LOAN DOCUMENT

PHOTOGRAPH THIS SHEET

AD-A254 948

LEVEL

1

INVENTORY

DTIC ACCESSION NUMBER

WL-TR-92-1060

DOCUMENT IDENTIFICATION

13 Jul 92

DISTRIBUTION STATEMENT

HANDLE WITH CARE

ACCESSION FOR

NTIS GRA&I

DTIC TRAC

UNANNOUNCED

JUSTIFICATION

BY

DISTRIBUTION/

AVAILABILITY CODES

DISTRIBUTION	AVAILABILITY AND/OR SPECIAL	
A-1		

DISTRIBUTION STAMP

DATE ACCESSIONED

DTIC QUALITY INSPECTED 1

DATE RETURNED

92 8 25 008

DATE RECEIVED IN DTIC

92-23574

402730

104 pgs

REGISTERED OR CERTIFIED NUMBER

PHOTOGRAPH THIS SHEET AND RETURN TO DTIC-FDAC

DTIC FORM 70A JUN 90

DOCUMENT PROCESSING SHEET

PREVIOUS EDITIONS MAY BE USED UNTIL STOCK IS EXHAUSTED.

LOAN DOCUMENT

WL-TR-92-1060

AN APPLICATION OF EXPONENTIAL NEURAL NETWORKS TO EVENT-TRAIN RECOGNITION

Peter G. Raeth
Passive Electronic Countermeasures Branch
Electronic Warfare Division

13 July 1992

Final Report for Period 5 Jan 90 - 1 Mar 92

Avionics Directorate
Wright Laboratory
Air Force Materiel Command
Wright-Patterson Air Force Base, Ohio 45433-6543

8 May 92 Final/5 Jan 90 - 8 May 92

An Application of Exponential Neural Networks to Event-Train Recognition

PE 62204F
PR 7633
TA 11
WU 13

Peter G. Raeth, Major, USAF, (513)257-5366

Passive Electronic Countermeasures Branch
Electronic Warfare Division
Avionics Directorate (WL/AAWP)
Wright Laboratory
Wright-Patterson AFB OH 45433-6543

WL-TR-92-1060

Approved for Public Release; Distribution is Unlimited.

The purpose of this project is to investigate neural networks for specific applications in passive electronic warfare (EW) involving restoration of deinterleaved pulse trains to their original broadcast form. The project took a generic event-train approach and focused on event-train recognition. It was determined that backpropagation neural networks did not represent a logistically supportable means of training. Gaussian radial basis functions were found to be far superior. This report is composed of three chapters: 1) summary of early experiments, 2) introduction to exponential neural networks, and 3) application to event-train recognition. Each chapter has its own references. We believe that the goal can be reached and that additional experiments, with greater data volumes, are warranted.

Neural networks, parallel processing, probability, backpropagation, basis functions, radial basis functions, Gaussian basis functions, electronic warfare, classification, pulse deinterleaving, threat alert, electronic combat.

95

Unclassified Unclassified Unclassified UL

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely Government-related procurement, the United States Government incurs no responsibility or any obligation whatsoever. The fact that the government may have formulated or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication, or otherwise in any manner construed, as licensing the holder, or any other person or corporation; or as conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

PETER G. RAETH, MAJOR, USAF
Deputy Air Force IR&D Manager
DCS/Science & Technology
HQ Air Force Materiel Command

PAUL S. HADORN, PhD, CHIEF
Passive ECM Branch
Electronic Warfare Division
Avionics Directorate
Wright Laboratory

JOHN H. SEWARD, JR, LT COL, USAF
Chief, Electronic Warfare Division
Avionics Directorate
Wright Laboratory

If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify WL/AAWP, WPAFB, OH 45433-6543 to help us maintain a current mailing list.

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

ACKNOWLEDGEMENTS

The author is grateful for the support provided by the following people and organizations:

Steve Gustafson and Gordon Little, University of Dayton Research Institute; and Daniel Zahirniak, USAF Wright Laboratory for their discussions on neural networks.

Debborah Ables, USAF Wright Laboratory, and Diana Owen, Headquarters Air Force Materiel Command, for their administrative support.

Rudy Shaw and Nicholas Pequignot, USAF Wright Laboratory, for their discussions on radar pulse sequence representation and deinterleaving.

Mark Thompson and James McCartney, SRL, for generating the radar training and test data.

University of Cincinnati College of Engineering Library and USAF Wright Laboratory Technical Library for the use of their literature search databases.

USAF Wright Laboratory Avionics Directorate for the use of their computing facilities.

Wright State University Library for the use of reference and loan materials.

(This page intentionally left blank.)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF FIGURES

INTRODUCTION

The purpose of this project is to investigate neural networks for specific applications in passive electronic warfare (EW) involving restoration of deinterleaved pulse trains to their original broadcast form. This is different from traditional bit-error detection/correction which relies on a prior knowledge of what the original bit stream looked like. In electronic warfare it is unlikely that such prior knowledge will be available.

The project took a generic event-train approach. Sequences of events were represented as a vector. Each vector element represented the length of time between the start of one event and the start of the next event in the sequence. For the case of pulse trains, the occurrence of a pulse is the event of interest. Known event sequences were used to train the neural network. The network then indicated the best match to an unknown test sequence, if such a match could be determined. Appendix A gives more details.

Backpropagation neural networks are currently the most popular for engineering applications. Thus, that network type was tried first. After a large number of experiments, we determined that backpropagation was not logistically supportable. Exponential neural networks were tried next with considerably more success.

Three major questions concerning neural networks of all types involve what training equation to use, what values to select for the parameters for these equations, and what is an appropriate network configuration. This project concentrated first on a review of relevant papers by various authors. A study was then performed on the backpropagation neural network to show how well it performed against a standard benchmark, the two-dimensional XOR problem. This involved having the network learn to recognize the XOR binary table. The study took the training equation and varied training equation parameter values and the network configuration. This study revealed the training reliability of the backpropagation neural network under varying conditions.

After a brief flirtation with adaptive resonance theory neural networks, the project settled on the exponential neural network as the best option. Several experiments were conducted to demonstrate the effectiveness of that network for this application.

This report is composed of three chapters. Chapter one summarizes the early experiments with backpropagation and offers an explanation of why that paradigm was abandoned for electronic warfare applications. Chapter two proposes an alternative to backpropagation, the exponential neural network. Chapter three applies the exponential neural network to the event train recognition and restoration problem. Each chapter has its own references.

CHAPTER ONE

SUMMARY OF BACKPROPAGATION EXPERIMENTS

By far, the most popular neural networks are those whose training is based on the backpropagation of output error. Particularly in the finance industry, there have been several cost-effective applications of this network type. A significant drawback of backpropagation is lack of predictable training time and resources. From the author's experience, for most new or modified tasks the backpropagation paradigm must be put through a series of steps unique to that task. These steps usually cannot be determined before the training session starts. The steps are typically determined "on the fly" by a human expert who must guide the training by hand using personal intuition and experience. This chapter summarizes a study of backpropagation neural networks performed during an attempt to transition that technology for defense applications in electronic warfare. We concluded that backpropagation exhibits unpredictable training and retraining characteristics. An additional conclusion is that, due to its unpredictability, backpropagation is generally not logistically supportable and is therefore not suitable for operational systems that have pre-determined reprogramming requirements.

The purpose of this research is to investigate neural networks for defense applications in electronic warfare. Event sequence

recognition and restoration was one application of particular interest. In this application, the neural network is trained on sequences of event types. For each sequence, the network is told at what times the event occurs and how long it lasts at each occurrence. The network's task is to examine a corrupted event sequence and decide if it is sufficiently close to any of the known sequences. The network must then match the corrupted sequence to a known sequence. Optionally, the corrupted sequence can be replaced by the known sequence to which it is most similar.

Event sequences may be represented in binary or analog form. A binary form is created by letting each element of a binary input vector represent a given time sample window. If an event starts during that window, a "1" is declared. Otherwise, a "0" is declared. (Only one event can happen at a time.) Figure 1-1a illustrates this idea. The analog version measures how much time elapsed between the start of one event and the start of the next event. It performs this measurement more accurately than to simply count the number of time windows between event starts. Figure 1-1b shows an example. The analog representation is preferred for this application because it is more accurate than simply counting time windows and because it greatly decreases the number of network inputs. The binary form requires an input for each time window whereas one analog input can stand for many time windows or a fraction of a window.

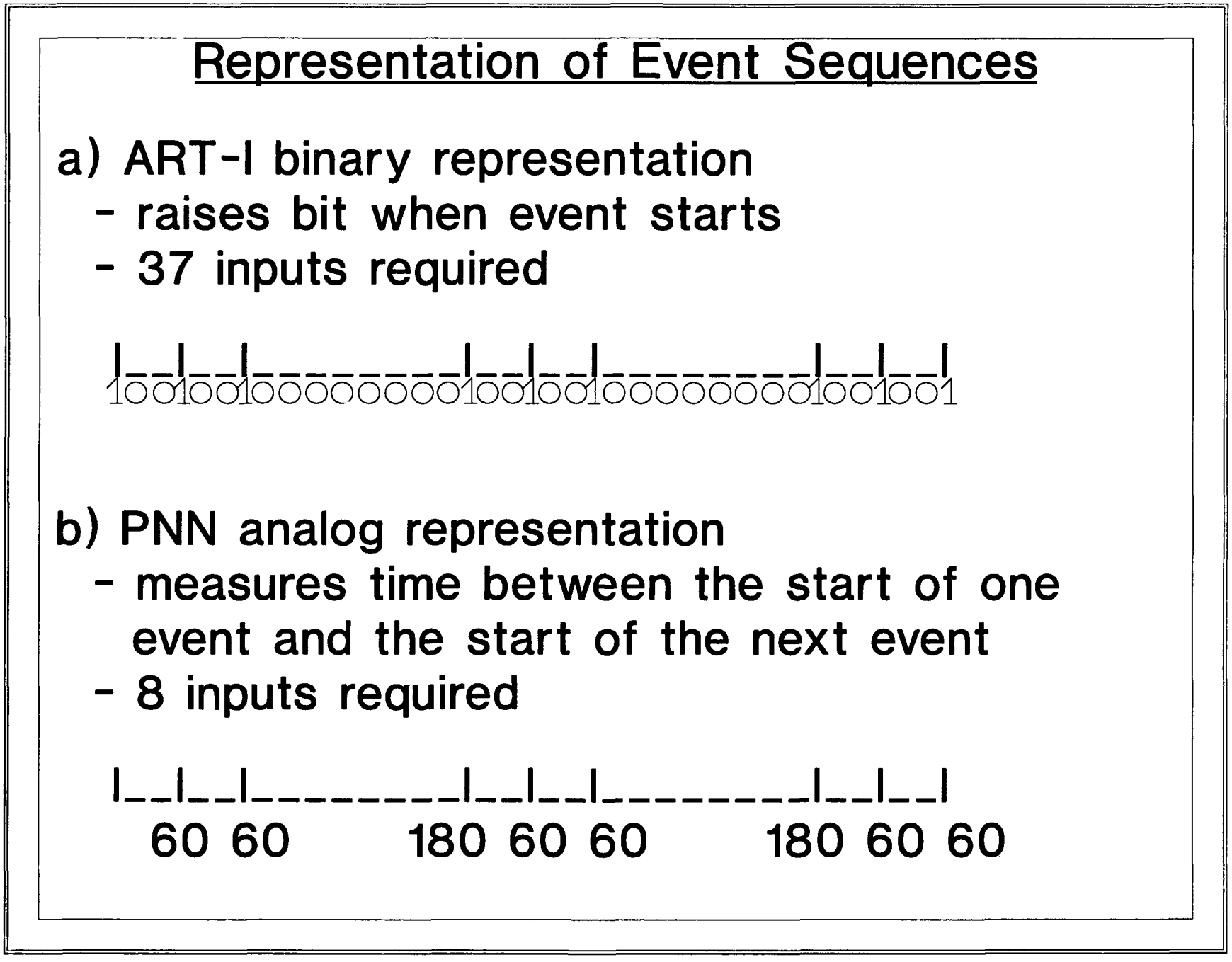

Figure 1-1: Representation of Event Sequences

The major question at the start of this research was how to evaluate the performance of a neural network relative to an application's objectives. Three choices must be made for neural networks of all types are:

1) the training equations
2) parameters of these equations
3) the appropriate network configuration

An experimental, as opposed to theoretical, approach to these choices was taken. We performed experiments involving up to 25,000 trials on selected networks.

Analyzing the data that resulted from these trials posed a difficult problem. Common two- and three-dimensional plotting methods served well at first. We ultimately decided to move on to four dimensional plotting methods. These methods offer a way of seeing the results of large numbers of trials in one picture.

Four dimensional plots show how variations of the values of global network parameters affect performance. A four dimensional plot illustrates the results derived from a function of three variables, $z = f(w,x,y)$, where Z is the function's result and W, X, and Y are the variables of the function. The values of these variables are systematically changed to produce Z and the function's plot (see

Figure 1-2). The forthcoming progression of examples will serve to illustrate the application of this plotting method to neural network performance analysis. These examples illustrate the best results achieved during a large number of experiments. (See Raeth [1], [2], [3], [4]; especially [2])

Basic Description of Backpropagation

Backpropagation, as described by Lippmann [5] and modified by Klimasauskas [6], is one popular method for training neural networks. This training method employs gradient descent techniques in an effort to reduce the network's error as it learns to perform the job it has been given (see Figure 1-3). Backpropagation is primarily affected by the following global network parameters:

Gain:	How much the network changes during each training iteration
Momentum:	How easily the direction of travel on the gradient can change
# Hidden Nodes:	Number of nodes in each layer between the input and output layers
# Hidden Layers:	Number of layers of hidden nodes
Initial Weight Setting:	The random starting state of the network
Order of Presentation:	In what order the network sees the training data

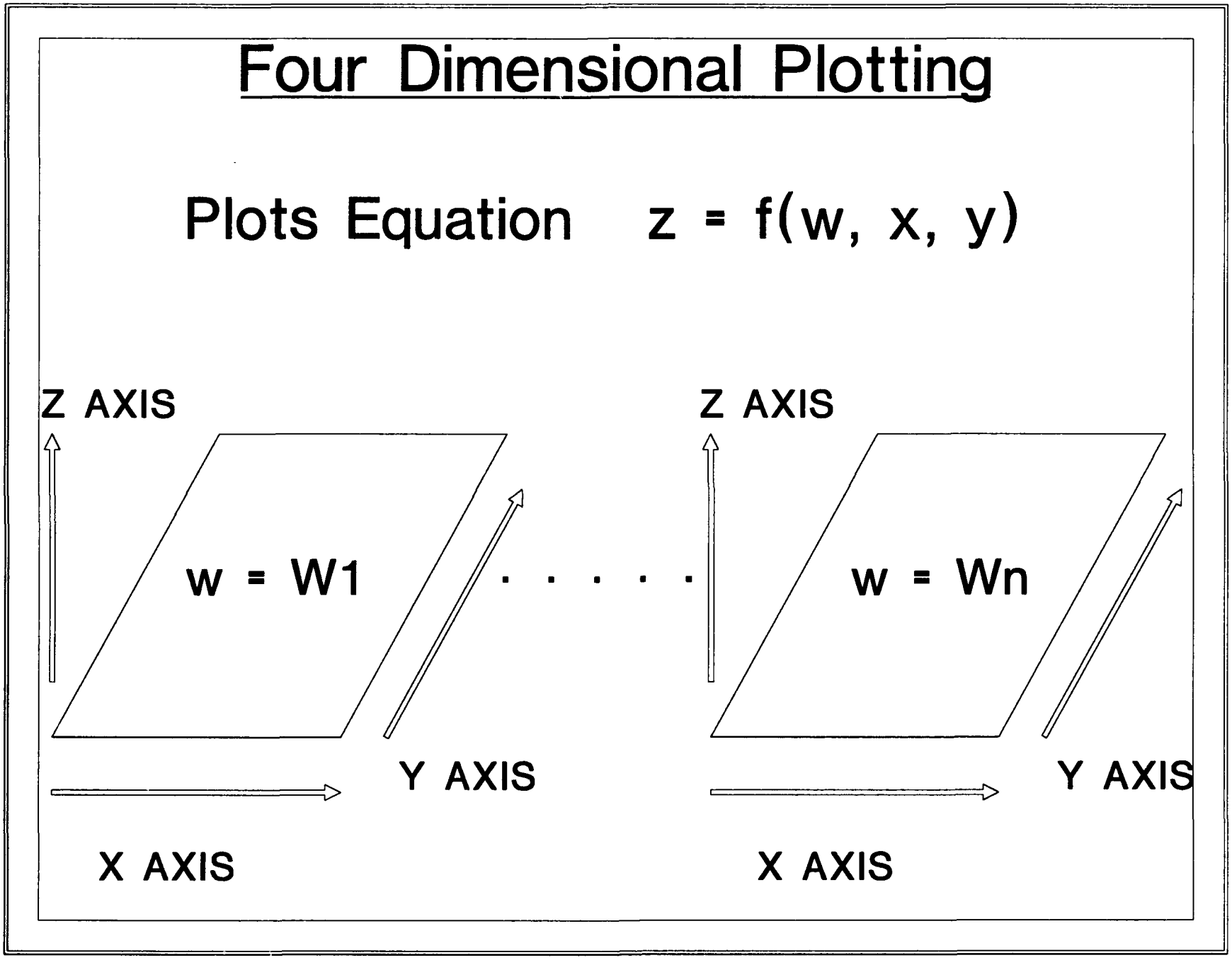

Figure 1-2: Four Dimensional Plotting

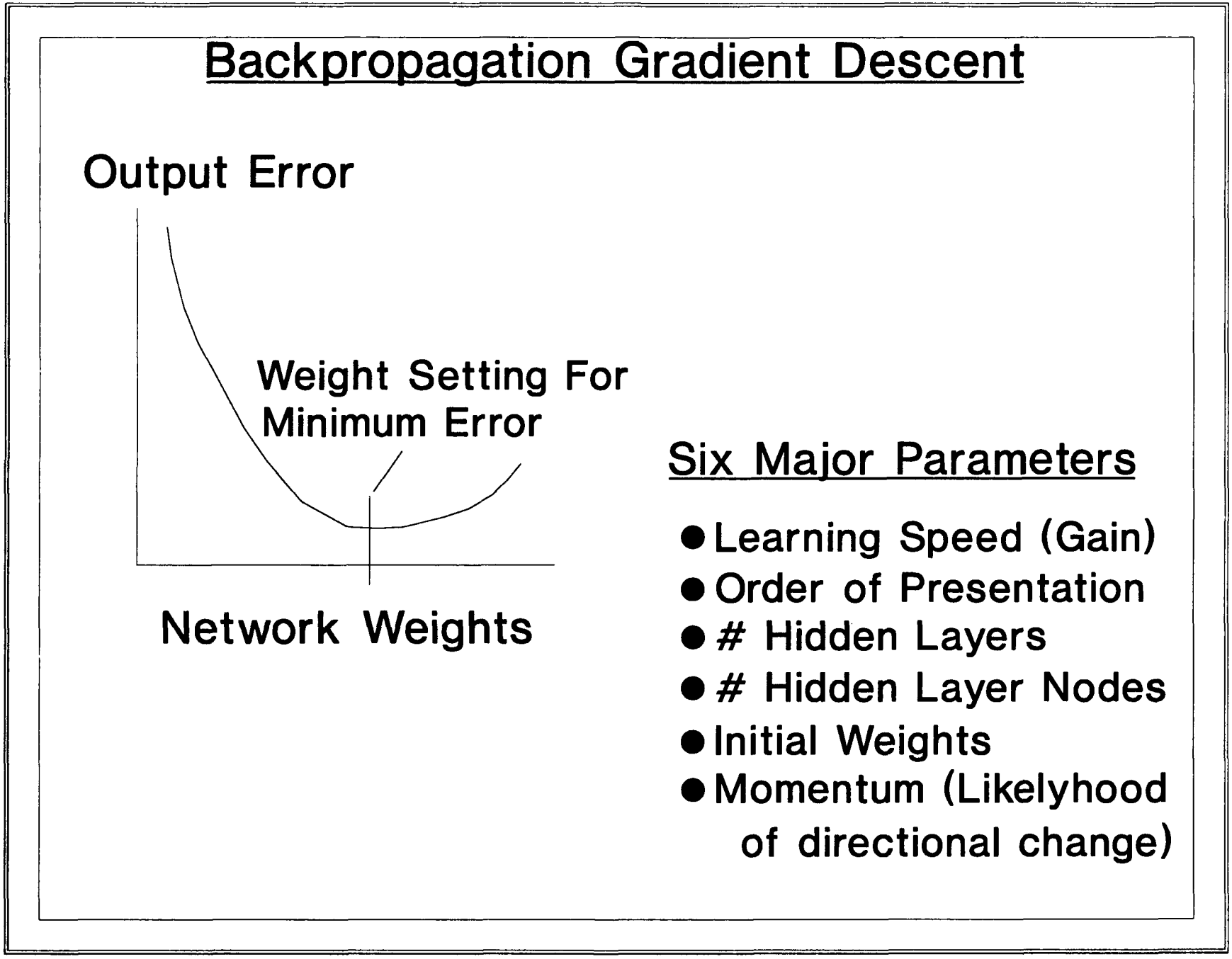

Figure 1-3: Backpropagation Gradient Descent

One goal for this network is that it should be reliable in its training. In other words, small changes in any of the primary parameters should not affect its ability to learn to perform the task it has been given.

A basic metric for testing the ability of a network to learn is the two-bit even parity (or XOR) metric illustrated in Figure 1-4. In this metric the parity bit is set to "1" if there is an odd number of "on" bits in the input. The parity bit is set to "0" if there is an even number of "on" bits.

As shown in the graph of the parity bit (Figure 1-4), there is no way to use a single straight line to separate the cases when the parity bit is set to "0" from the cases when it is set to "1." Therefore, the cases are said to be linearly inseparable or disjoint. The metric requires the neural network to accept two bits as input and to learn to generate the correct parity bit as output. The XOR metric is similar in many ways to the binary form of the event sequence recognition problem. The neural network has to identify unique binary patterns and generate an appropriate binary response.

Two-Bit Even Parity (XOR) Metric

Network Inputs		Required Output
#1	#2	Parity Bit
0	0	0
0	1	1
1	0	1
1	1	0

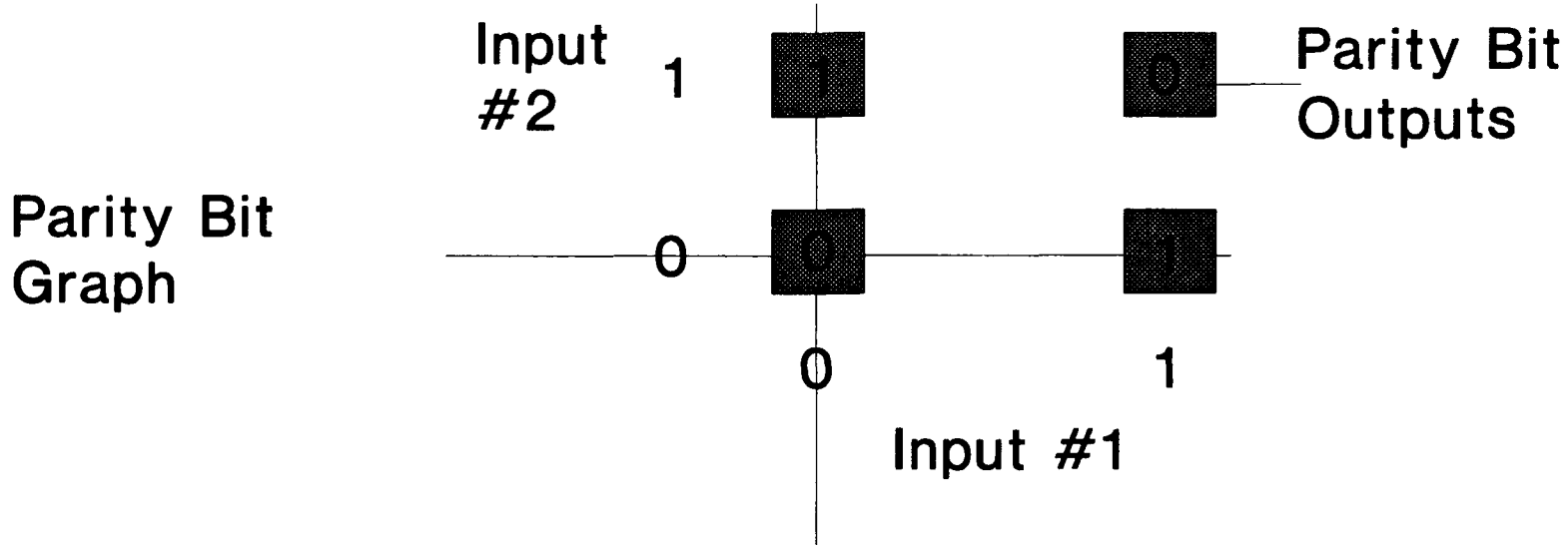

Figure 1-4: Two-Bit Even Parity (XOR) Metric

Experiments With Three Layers

Figure 1-5 shows two backpropagation neural network architectures that can learn to perform the XOR metric. The first architecture has three layers: an input layer, a transformation (or hidden) layer, and an output layer. Two bits are inserted at the input. The network must learn to generate a "1" or "0" at the output as appropriate for the input. For a first experiment, let's hold all parameters constant and just vary Gain from 0 to 4 in 200 steps (Momentum = 0, Number of Hidden Nodes and Layers = 1 the same weight initialization randomly chosen from a uniform distribution between -0.1 and +0.1 inclusive, and the same list of randomly selected input vectors). For each value of Gain, we will train the network to perform the XOR metric.

The resulting plot in Figure 1-6 has Gain along the X axis and number of exposures required along the Z axis. The plot shows how many randomly selected training data exposures were required for the network to learn the XOR metric for each setting of Gain. (An exposure is one vector chosen randomly from the list of four XOR vectors. During a single exposure, the network is allowed to train for that vector only. For each attempt to learn the entire XOR metric, all network variables are held constant.) For very small values of Gain the network either did not learn at all (the top of the graph) or learned very slowly. For medium values of Gain the network was very quick to learn. For high values of Gain, the

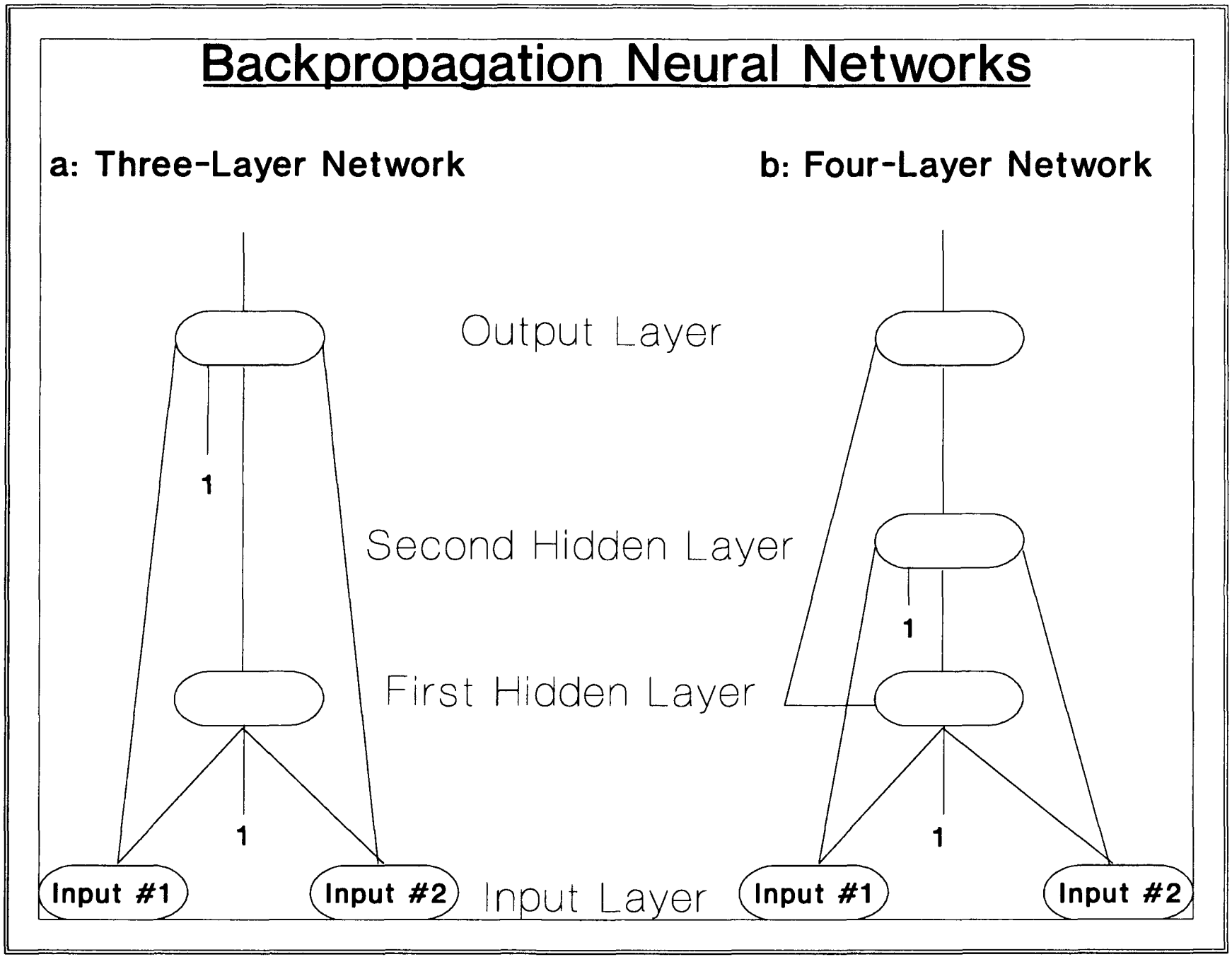

Figure 1-5: Backpropagation Neural Networks

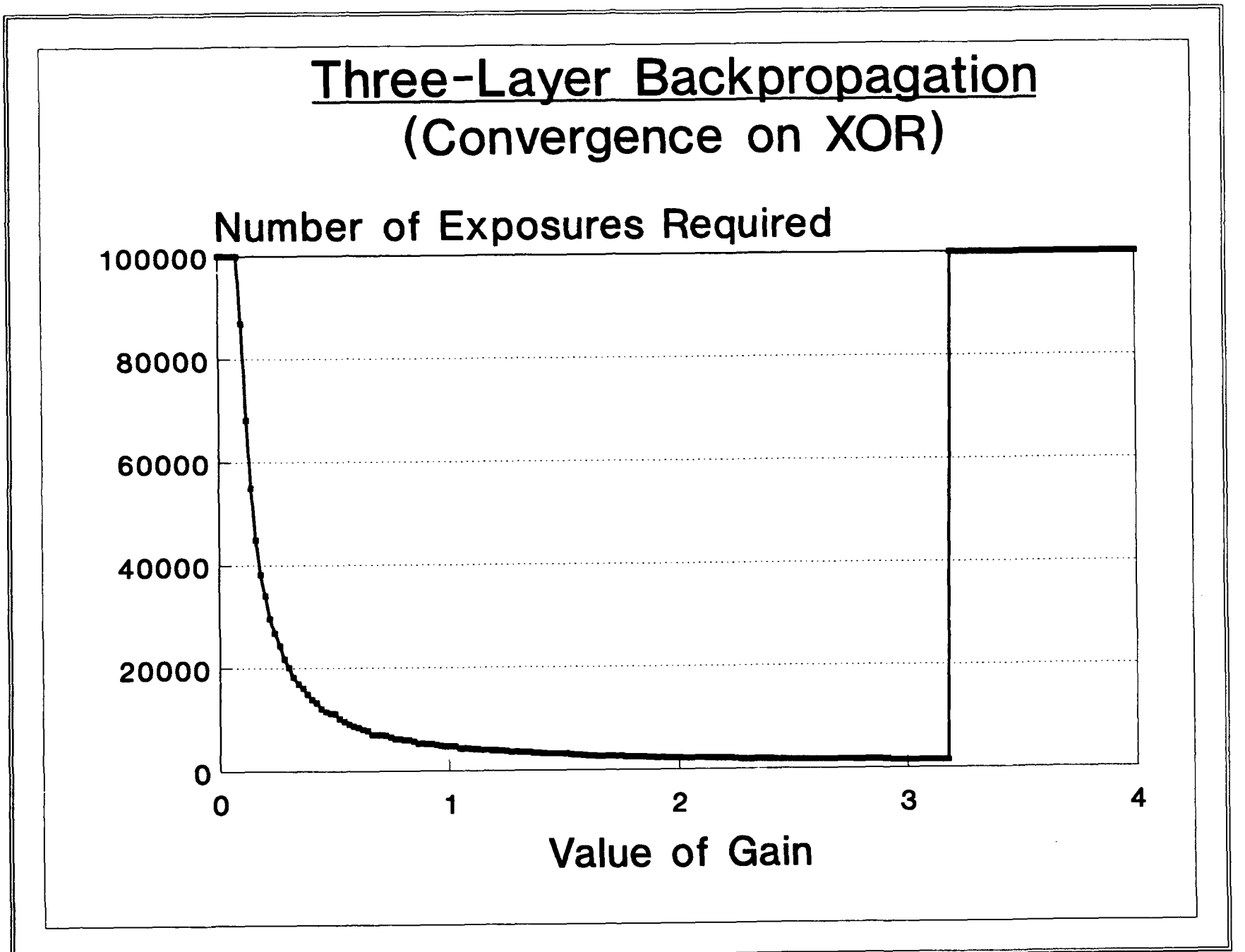

Figure 1-6: Three-Layer Backpropagation Convergence on XOR

network was, again, unable to learn. (Training stopped at 100000 exposures if the network had not yet learned the entire XOR metric.)

Lets move the plot up a dimension by varying Gain from 0 to 4 in 50 steps and Number of Hidden Nodes from 1 to 10 in 10 steps (500 trials in all).

Gain is still on the X axis and Number of Hidden Nodes is on the Y axis. The number of exposures required is shown again on the Z axis. As shown in the plot of Figure 1-7, as Gain and Number of Hidden Nodes increases, a small change in their values causes the network to switch rapidly from fast training performance to very slow or non-existent training performance. For high values of Gain and Number of Hidden Nodes, the network loses its predictable training performance.

For logistical supportability, it is more important for the network to have predictable training performance than fast performance. In our experience, it is often the case that networks of this type exhibit their fastest training performance in regions of unpredictable performance.

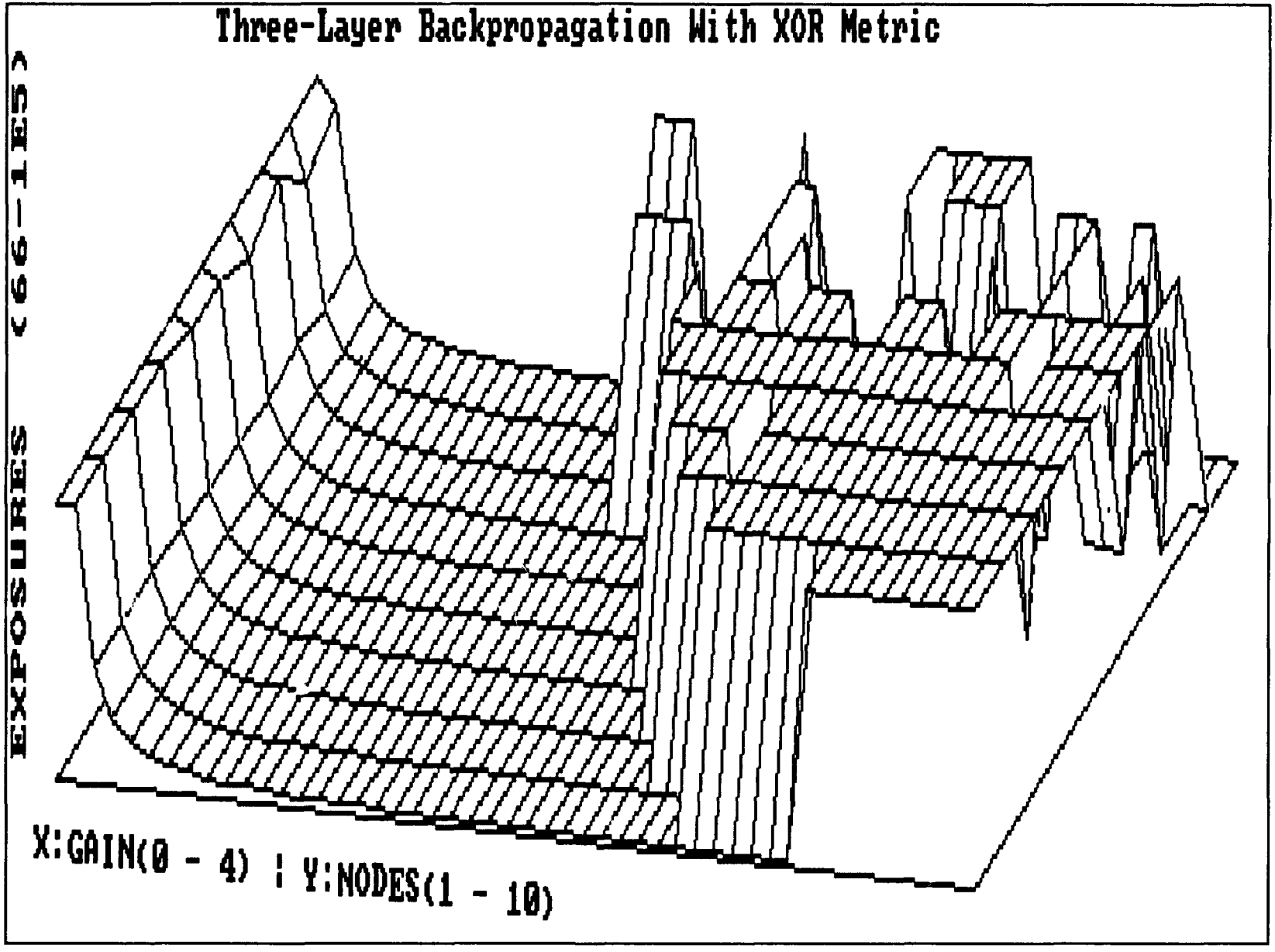

Figure 1-7: Three-Layer Backpropagation With XOR Metric

Experiments With Four Layers

In a search for greater reliability, we ran that same experiment with the four-layer architecture shown in Figure 1-5b.

As shown in Figure 1-8, this somewhat more complex architecture has significantly increased the reliability of the network. Only for small values of Gain and one hidden node in each layer does the network not learn reliably. Having obtained a network that appears to be very reliable, it was worth our while to go on to a more lengthy experiment with the four-layer architecture.

In this next experiment, Gain was varied between 0 and 4 in 50 steps, Number of Hidden Nodes between 1 and 50 in 50 steps, and Momentum from 0 to 1 in 10 steps. The resulting 25,000 trials were used to generate an animated color coded surface that shows how the change in Momentum causes the network's learning ability to evolve. Each frame of the animation shows how learning ability is affected by variations of Gain and Number of Hidden Nodes for a given setting of Momentum. Gain is plotted along the X axis, Number of Hidden Nodes along the Y axis, Number of Exposures Required along the Z axis, and Momentum along the W axis. The W axis is the time axis in that Momentum is increased sequentially for each animation frame (remember Figure 1-2). The computer linearly interpolates between each calculated frame to produce the animation.

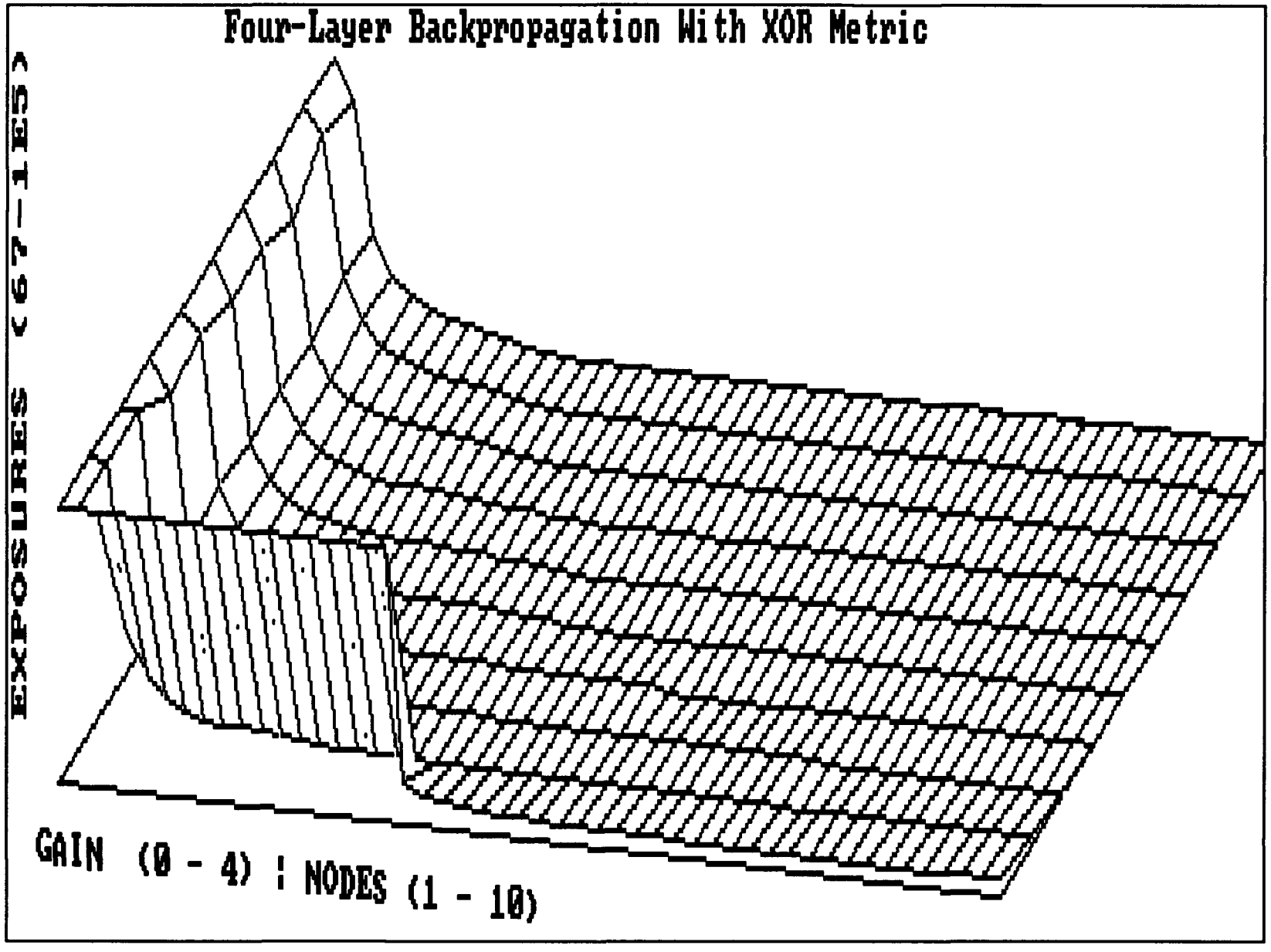

Figure 1-8: Four-Layer Backpropagation With XOR Metric

Figure 1-9 shows four frames of this plot.

The network's ability to learn the XOR metric slowly degrades as Momentum increases. With low values of Momentum there is highly reliable learning except at high values of Gain and Number of Hidden Nodes. Training ability collapses at high values of momentum (See Figure 1-9).

The animation shows other information as well. There is a cyclical nature to the edge of the area where the network does not learn. This is not an artifact of the animation. This cyclical action is present in the raw data. Something like this would never be evident if you just looked at the raw data or discrete frames. Increasing action in the animation indicates an increasing rate of change in performance as long as the frame rate is held constant. By speeding up or slowing down the animation it is possible to see data aggregations at higher or lower levels of detail.

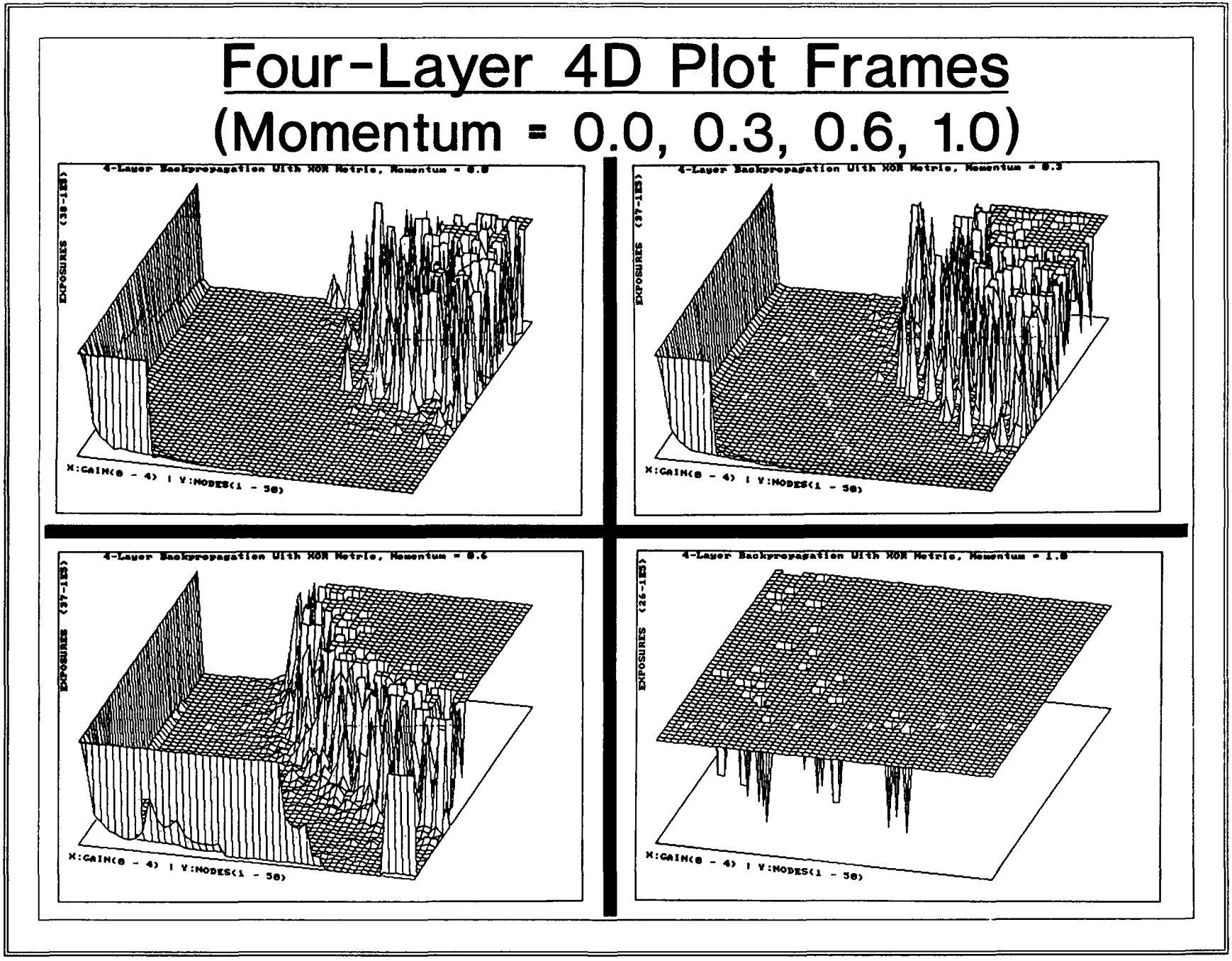

Figure 1-9: Four-Layer 4D Plot Frames

Determinations

After performing these and many other experiments with backpropagation and reading of other practitioners' experiences with that training method, a negative conclusion relative to the target application of this research seemed inescapable. This conclusion drives efforts in other facets of this research and the author's reviews of contractor research reports. The XOR metric is a relatively trivial problem when compared with the application introduced here and other potential uses for neural networks in electronic warfare.

Based on observations of the experiments summarized in this chapter, there are several factors that prevent networks trained via backpropagation from being logistically supportable:

a) A large number of variables are involved in getting these networks to train.
b) Their unpredictable training characteristics prevent their training in a known amount of time and resources.
c) The research community has not established that backpropagation can meet operational defense reprogramming requirements.
d) The level of education and experience required to deal effectively with backpropagation is not likely to be found in logistic support centers.

Rogers et al. [7] have noted several factors that are also a cause for concern:

a) There is no general theory for determining apriori the appropriate architecture and variable settings for a backpropagation network based on the problem to be solved, p60.
b) What rules of thumb exist are proven only for two-class problems or for very limited cases, p60.

c) There are many backpropagation learning algorithms but which one will permit the network to converge for a given task is open to question, p58. (The commercial neural network development tool NeuralWorks Professional II+ currently has 8000 versions of backpropagation [8]. These various versions have been developed by practitioners attempting to speedup backpropagation training.)

d) There are secondary variables such as numerical accuracy that must be considered, p59. (The author has found that the random sequence generated by a given random number generator is another of these secondary variables.)

Conclusion on Backpropagation

The most important point is that, at the present time, backpropagation networks are too unpredictable during training. These networks are still a matter of basic research and are not ready for application in fielded equipment. Therefore, this author has abandoned backpropagation in favor of neural networks that learn in one pass through the training data, or at least in a known number of passes. These networks have their own advantages and disadvantages. But, they have proven much more successful for operational application than backpropagation networks. The networks in question are the Adaptive Resonance Neural Network (Carpenter [9], Grossberg [10], Lippmann [5]), the Probability Neural Network (Specht [11], Maloney [12]), the more general class of Radial Basis Neural Networks (Zahirniak [13]), and the Stretching and Hammering Neural Network (Gustafson et al. [14]). These networks are not iteratively trained via an unknown number of passes through the training data whereas backpropagation networks are. Also unlike backpropagation, the resources they require for

training are predictable.

The research with neural networks that train in a known number of passes through the training data has shown that they have strong promise for overcoming the growing information processing and reprogramming bottlenecks found in operational defense systems. They are logistically supportable and have predictable training and performance characteristics.

Closing Remarks on Backpropagation

Because of the unresolved issues that negatively affect the predictable training and retraining of backpropagation neural networks, this author has abandoned all attempts to transition such networks into defense operational systems that have pre-determined reprogrammability requirements. While this author feels that backpropagation neural networks are still very much a matter of basic research, continued work in this area may yet yield transitionable results. The answer may lie in a combination of the ideas behind backpropagation and single-pass radial basis function neural networks. In this regard, the work of Buntine and Weigend [15] as well as Weymaere and Martens [16] appears to be very promising. Another approach, combining radial basis networks with learning vector quantization, has been undertaken by Burrascano [17].

Neural networks that train in a predictable number of passes through the training data using predictable resources show promise for overcoming the information processing and reprogramming bottlenecks that result from defense operational requirements.

Chapter One References

[1] Raeth, Peter G. "3-D Surface Maps for Neural Network Performance Evaluations," Dayton SIGART Aerospace Applications of Artificial Intelligence Conference, Oct 89

[2] Raeth, Peter G. "Event-Train Restoration Via Backpropagation Neural Networks," Available through DTIC/NTIS # AD-A216-308, Dec 89

[3] Raeth, Peter G. "Using 3D Surface Maps to Illustrate Neural Network Performance," Proc: IEEE National Aerospace and Electronics Conference (NAECON), May 90

[4] Raeth, Peter G. "An Experiment with 3D Surface Maps to Illustrate Neural Network Performance," Proc: IEEE/INNS International Neural Network Conference, Jul 90

[5] Lippmann, Richard P. "An Introduction to Computing with Neural Nets," IEEE ASSP Magazine, Apr 87

[6] Klimasauskas, Casimir C. "Neural Networks: A Short Course," PC AI Magazine, Nov/Dec 88

[7] Rogers, Steven; Matthew Kabrisky; Dennis Ruck; and Gregory Tarr An Introduction to Biological and Artificial Neural Networks Bellingham, WA: Optical Engineering Press, 1991

[8] Klimasauskas, Casimir; John Guiver; and Garrett Pelton Neural Computing Pittsburgh, PA: NeuralWare, Inc; 1989

[9] Carpenter, G.A. and Grossberg, S., "ART2: Self-Organization of Stable Category Recognition Codes for Analog Input Patterns," Applied Optics, pp 4919-4930, 1987

[10] Grossberg, S., "Contour Enhancement, Short Term Memory, and Constancies in Reverberating Neural Networks," Studies in Applied Mathematics, 213-257, 1973

[11] Specht, D. F. "Probabilistic Neural Networks for Classification, Mapping, or Associative Memory," Proc: International Conference on Neural Networks, 1988

[12] Maloney, P.S. "Use of Probabilistic Neural Networks for Emitter Correlation," Internal Report, Lockheed Missiles and Space Co, 89

[13] Zahirniak, Daniel R. Characterization of Radar Signals Using Neural Networks Dayton, OH: Air Force Institute of Technology, Thesis # AFIT/GE/ENG/90D-69, also appears in DTIC/NTIS, Dec 90

[14] Gustafson, Steven; Gordon Little; John Loomis; and Todd Puterbaugh "Stretching and Hammering Algorithm for Data Representation and Generalization," Internal Paper; Dayton, OH: University of Dayton Research Institute; Submitted to: Communications in Applied Numerical Methods, Mar 91

[15] Buntine, Wray L. and Andreas S. Weigend "Bayesian Backpropagation," Unpublished report undergoing journal review, NASA Ames Research Center, Jun 91

[16] Weymaere, Nico and Jean-Pierre Martens "A Fast and Robust Learning Algorithm for Feedforward Neural Networks," Neural Networks, pp 361-369, 1991

[17] Burrascano, Pietro "Learning Vector Quantization for the Probabilistic Neural Network," IEEE Transactions on Neural Networks, pp 458-461, Jul 91

(This page intentionally left blank.)

CHAPTER TWO

DESCRIPTION OF EXPONENTIAL NEURAL NETWORKS

Voting is one method used in expert systems to deal with uncertainty. Evidence on the validity of one or more decisions is collected. Each piece of evidence then casts a "vote" in favor of the defended decision(s). Thus, evidence in favor of various alternatives is accumulated. The power of each vote depends on the applicability of the associated evidence. Applicability controls the contribution each piece of evidence makes toward any specific decision.

A difficult problem in voting is specifying the contribution of each vote. One method of specification uses the exponential neural network (ENN) discussed in this chapter. The ENN is a method for implementing voting mechanisms in expert systems, a form of knowledge-based system, using neural network technology, another form of knowledge-based system. Therefore, a link may exist between traditional expert systems and neural networks. Neural networks provide an O(1) implementation of a well-known decision support method.

An advantage of the ENN, as compared to some other types of neural networks, is its ease and predictability in training. Additionally, the ENN has only one parameter to be set by the user, the desired generalization. This chapter illustrates the theory

and implementation of the ENN. An example application to pattern recognition is also given as are references to related work. The ENN is member of the more general class of radial basis function (RBF) neural networks.

Basic Principles

Supervised-learning neural networks may be said to accumulate evidence toward classification decisions in that samples are collected to represent a certain class or classes. These samples contribute evidence toward a decision on which class is represented by a test input of unknown class.

Because the *samples* (evidence) can never be completely descriptive of all possible classes (decisions), some way is needed to deal with the resulting uncertainty. In all types of knowledge-based systems, the basic requirement is to have an objective (and accurate) numeric procedure for arriving at a decision given the attendant uncertainty [1]. It is possible to assign a certainty factor valued between zero and one to indicate the strength of individual pieces of evidence according to how well each supports a given decision. The evidence can then be combined to determine the relative strength of each decision. Thus, the certainty (or strength) factor can be used as a type of vote for the various decisions.

Other forms of reasoning under conditions of uncertainty have been described by Groothuizen [2] and by Sudkamp et al. [3]. The major forms they mention are fuzzy logic [4], Dempster-Shafer reasoning [5], and probability theory [6]. Neural network methods could be added to those forms as is clearly seen in the work of Specht [7] and Ruck et al. [8]. This chapter argues that neural network methods are applicable to the reasoning problems faced in older forms of knowledge-based systems, namely expert (or advisory) systems.

In applying voting methods to reasoning problems involving uncertainty, one can begin by combining the concepts behind approximate and plausible reasoning as defined by Groothuizen [2]. He defines approximate reasoning as deductive inference using uncertain or imprecise premises. This basic definition leads to the assumption that some training samples may be misclassified, that more than one class can be indicated by the same sample, and that there is some level of imprecision in the values of the sample elements (features). To compensate, several samples for each class are collected, but it is still necessary to assume that errors and multiples are relatively small. Groothuizen defines plausible reasoning as the derivation of uncertain conclusions using certain premises. Since it is assumed that classes are incompletely specified by the samples and are inaccurate to some degree, conclusions based on new test inputs are uncertain. However, the ENN calculations that lead to a decision are based on the

assumption that the samples are accurate and correctly classified even in the case of multiple classes for the same sample. This assumption can be removed by discounting the sample elements with a weighting factor. Discounts are predicated on the expected inaccuracy of the method of gathering and reporting element values.

Voting mechanisms have been discussed by Blockley and Baldwin [9] in that they propose the combination of evidence for the same conclusion. "In any knowledge base it may be possible to obtain support for a conclusion from any number of sources." The ENN gives a full vote or fraction of a vote per sample per class based on the Euclidean distance of each sample from the test input. The votes from the samples are summed on a class-by-class basis to obtain a relative evidence factor supporting each class. Note that this scheme does not exclude the possibility that a test input belongs to other than the top-ranked class. To reject lower ranked classes would be a negation by failure (insufficient evidence), which has no logical justification, according to Shepherdson [10]. The ENN does not reject class alternatives. It only rank-orders the alternatives. The summation of votes also does not rely on a closed-world form of knowledge-representation as does probability theory's requirement that all probabilities must sum to one. Specht's [7] probabilistic neural network (PNN) does implement this closed-world view. The ENN can be evolved into the PNN.

The voting approach also finds support in the comments of Tou and

Gonzalez [11, p5], "... the problem of pattern recognition may be regarded as one of discriminating the input data, not between individual patterns but between populations ... ." They also state, "Human recognition is in reality a question of estimating the relative odds that the input data can be associated with one of a set of known statistical populations that depend on our past experience and which form the clues and the apriori information for recognition." This chapter takes their term "population" to refer to the body of evidence collected on specific classes, the matrix of class samples. Thus, classification under conditions of uncertainty attempts to "fit" the test input to the most likely class or population of samples.

Implementing the Exponential Neural Network

The ENN's implementation follows directly from the stretch and hammer neural network proposed by Gustafson et al. [12]. This network was designed for n-dimensional smooth-surface interpolation based on random samples. However, since classification is a special case of interpolation, it is not difficult to modify the network for classification. Besides the PNN, Specht [13] has developed a neural network for regression which also performs n-dimensional smooth-surface interpolation.

The theory behind the stretch and hammer neural network was reduced to practice by Raeth et al. [14]. Their report presents exhaustive

network implementation and testing detail. Briefly, the stretch and hammer neural network combines a least-squares hyper-plane fit to the training samples with Gaussian functions centered at the training samples. The sum of these Gaussians and the least-squares plane gives the height of the surface in Euclidean space at the coordinates of the training samples. The fundamental equation executed by the stretch and hammer neural network is:

$$z = \mathbf{g}\mathbf{b} + \mathbf{t}\mathbf{q} + q_0$$

where

b is a vector of Gaussian coefficients
q is a vector of least-squares coefficients, $(q_1, q_2, \ldots, q_n)^T$ (T refers to the vector transpose)
q_0 is the least squares plane z-axis intercept
g is the row vector of Gaussian training functions calculated at the test point
z is the scalar value of the surface at the coordinates of the test point
t is the test point taken as a row vector
n is the number of elements (dimensions) in the test and sample inputs

The following paragraphs describe the arrangement of the stretch and hammer neural network as used by the exponential neural network. Only the Gaussian part of the stretch and hammer neural network is used and the Gaussian coefficients are all set to one. Essentially, the ENN replaces the least-squares plane with the zero plane.

Classical Gaussian radial basis functions for n-dimensional spaces are used by the ENN to set the voting contribution of the samples. Each sample has a Gaussian function centered at the sample's position in Euclidean space. The vote contribution equation employed by each sample is given below. This equation reflects the ENN's current implementation. Any basis function could be used with appropriate modification to the supporting mathematics.

$$c_k = e^{\frac{-\sum_{i=1}^{n} (t_i - s_{ki})^2}{2\sigma_k^2}}$$

where σ_k^2 is the variance of the k'th sample's Gaussian function
e is the base of natural logarithms
n is the number of elements (features or dimensions) in each sample or test input
t_i is the i'th element in the test input
s_i is the i'th element in the k'th sample
c_k is the vote contribution of the k'th sample

Figure 2-1 illustrates the voting contribution from a single one-element sample whose Gaussian function's $2\sigma^2 = 0.05$. The sample is located at s = 0. Note that the voting contribution of the sample is equal to one whole vote at the sample's location and decreases as the distance from the sample increases. This is an indication that, from the sample's point of view, there is less evidence to support the test input's membership in the sample's class as the test input's Euclidean distance from the sample increases. The amount of decrease with distance is controlled by $2\sigma^2$. The smaller the value, the faster the decrease. However, the peak value at the sample location is always equal to one.

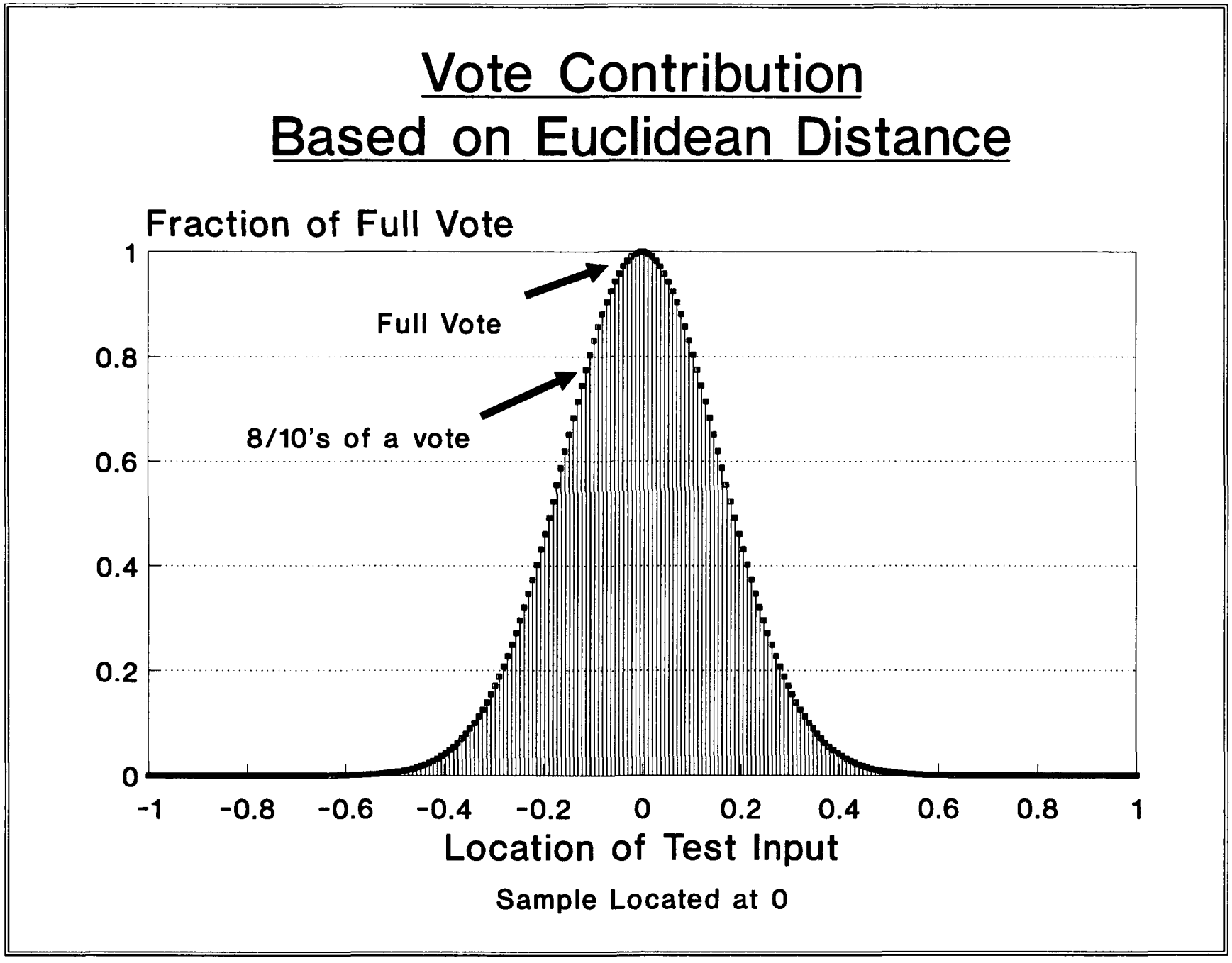

Figure 2-1: Vote Contribution Based on Euclidean Distance

Once the contribution curve has been established for each sample, the specific contribution of each sample to the body of evidence that supports the classification of a given test input may be calculated. This contribution is the c_k shown above. The contributions of all samples are summed on a class-by-class basis to obtain the total evidence that the test input belongs to a specific class. This summation is shown in the equation below.

$$\varepsilon_j = \sum_{k=1}^{m} c_k$$

where c_k is the contribution from sample k of class j as described above
m is the total number of samples lending evidence to class j
ε_j is the evidence that the test input belongs to the j'th class

For example, consider the two-sample single-dimension case illustrated in Figure 2-2. The samples are located at s = 0.0 and s = 0.25. Both their Gaussian functions have $2\sigma^2 = 0.05$. As the value of t, the test input, moves from -1.0 to +1.0, the evidence that it belongs to the class represented by the two samples follows the ε curve.

An ε curve is generated for each class, and the value of ε at the test input is calculated. Generally, the class with the highest value of ε is selected as the class to which the test input most likely belongs. Kabrisky and Rogers [15, p10] have suggested a variation on the idea of accepting the highest class output from a classification system as the class of choice. They calculate a

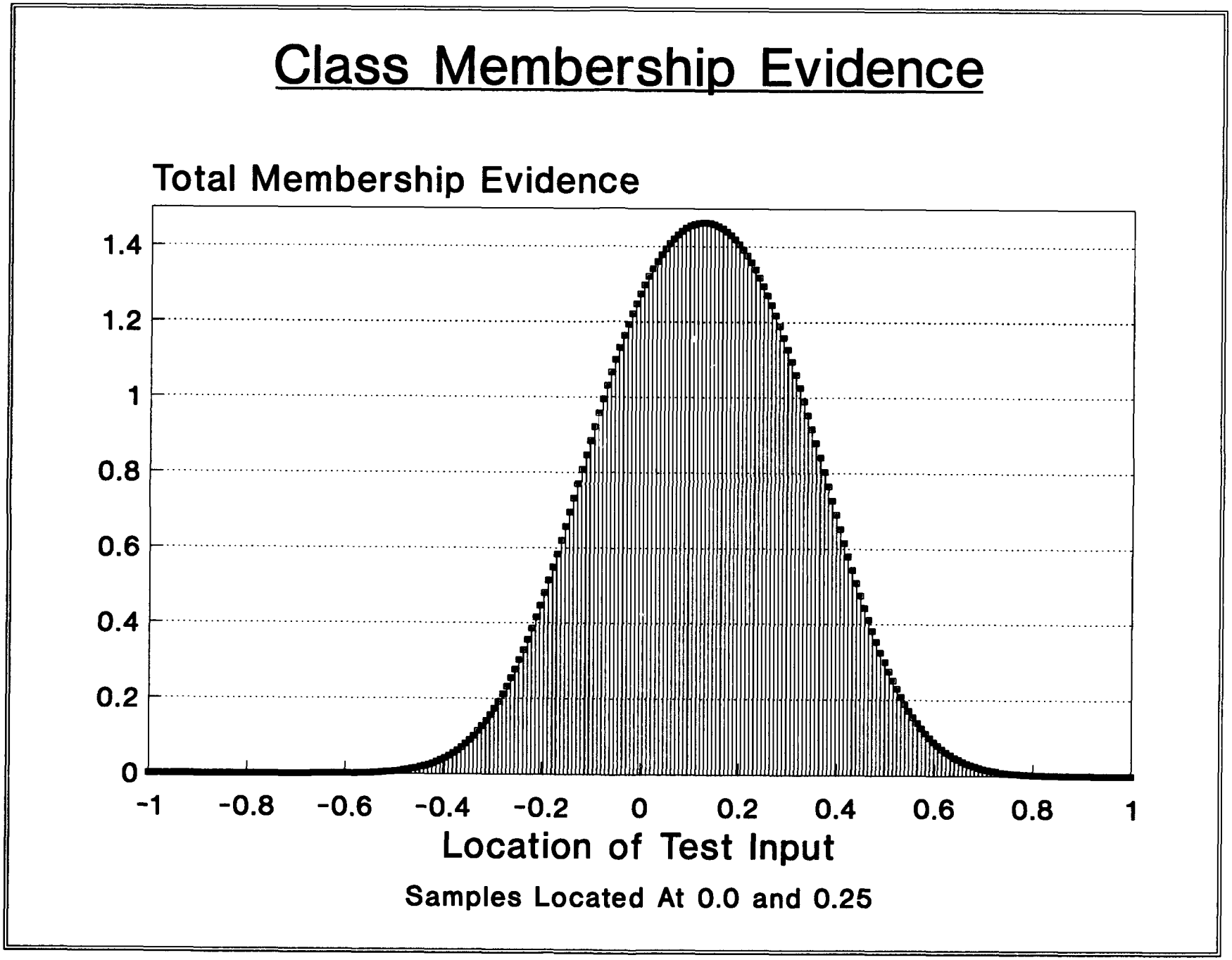

Figure 2-2: Class Membership Evidence

figure-of-merit (FOM) that accounts for ambiguity. The closer a class FOM is to one, the higher the ambiguity, as shown in the equation below.

$$FOM_h = \frac{\varepsilon_h}{\varepsilon_{h-1}}$$

where FOM_h is the figure-of-merit for the highest ranked class
ε_h is the evidence for the highest ranked class
ε_{h-1} is the evidence for the next-to-highest ranked class

Another variation is to not accept any classification for which the evidence is not above a specified threshold. Both variations can be combined.

It can be seen from the above description that the supporting evidence curve for a given class is adjusted automatically based on the positions of the training samples. As new samples are added to the system, the evidence curve is adjusted further so that the impact of the new samples on ε may be determined. Thus, training the ENN is simply a matter of setting $2\sigma^2$ and providing the samples to the network. A question remains, however, on how to set $2\sigma^2$ for each sample's Gaussian function.

There are several techniques for setting $2\sigma^2$ in radial basis function neural networks. Zahirniak [16, p23] suggests three methods: 1) set to some positive constant, 2) set according to a distance metric that measures the distance between samples, and 3) set so that samples from one class do not respond too strongly to

inputs of another class. Another method is to set $2\sigma^2$ so that each sample's Gaussian value is at a specified level at some point (e.g. the half-way point) between itself and the nearest neighbor sample. This method derives from Zahirniak's second method and limits the generalization capability of each sample according to how close it is to its nearest neighbor, independent of the nearest neighbor's class. The $2\sigma^2$ can be set not only sample-by-sample but also dimension-by-dimension within each sample.

Gustafson et al. (reported in [14]) have developed the columnar diagonal dominance method for setting $2\sigma^2$ for the individual samples' Gaussian functions. This method not only shapes the ε curve depending on the number and position of all the samples but also permits the extension of the ENN to the stretch and hammer neural network's smooth-surface interpolation capability.

Recall that in ENN training, Gaussian radial basis functions are centered at each training point. The Gaussian function outputs may be placed in matrix $\mathbf{F}$. Matrix $\mathbf{F}$ is an m' by m' square matrix where m' is the overall total number of samples. Each $2\sigma^2$ is adjusted so that the matrix of Gaussian equations that results is diagonally dominant by columns. Columnar diagonal dominance requires the absolute value of the sum of the off-diagonal elements of any matrix column to be less than the absolute value of the diagonal element in that column.

Each column of the **F** matrix represents a Gaussian function centered at a given training point. Each element of each column represents the output of the column's Gaussian function at the element's training point. The value stored at each matrix element is the Gaussian function centered at the column's training point as evaluated at the training point indicated by the element. For example, column #1 indicates the Gaussian function for training point #1. The first element of column #1 indicates the output of the Gaussian for the first training point as calculated at the first training point, the second element of column #1 indicates the output of the Gaussian for the first training point as calculated at the second training point, and so on. Thus, column #1, element #2 of matrix **F** stores the output for the first training point's Gaussian function calculated at the second training point. Note that the diagonal elements of matrix **F** always equal unity and that the off-diagonal elements are always positive but less than unity.

For many choices of $2\sigma^2$ the matrix **F** is singular. A singular **F** matrix results in a final stretch and hammer neural network that does not fully represent the training data. Thus, interpolation is less accurate. (Matrix **F** can be viewed as the left hand side of a system of equations. According to Press et al. [17, p28], A degenerate set of equations is called 'singular.' If one or more of the equations is a linear combination of the others, row degeneracy exists. If all equations contain certain variables only in exactly the same linear combination, column degeneracy exists.)

A solution to the problem of degenerate sets of equations is to choose each sample's $2\sigma^2$ to ensure that the matrix $\mathbf{F}$ can never be singular. A computationally efficient approach is to select the variances (σ^2) so that matrix $\mathbf{F}$ is diagonally dominant by columns. Physically, columnar diagonal dominance occurs when the variances are sufficiently narrow that the Gaussian functions' values at neighboring (and, therefore distant) points are small. Since extremely narrow variances would result in a pincushion interpolating surface having poor smoothness, it is reasonable to make the variances as large as possible while maintaining columnar diagonal dominance and classification accuracy. This condition is achieved using a short iterative procedure. This procedure sets each column's variance to a small value and then sums the off-diagonal elements in that column (in the case of Gaussian functions the off-diagonal sum is always positive). If the summation is not less than some value less than unity within some margin of error (say 0.001), then the variance is appropriately modified and the procedure is repeated. The procedure is performed for each column in the $\mathbf{F}$ matrix.

Consider the ε curves illustrated in Figures 2-3 and 2-4. Class 1 has samples located at -0.75 and -0.50. Class 2 has samples located at +0.5 and +0.75. Initially, as shown in Figure 2-3, $2\sigma^2$ for each sample's Gaussian function has the value 0.005. Then, as graphed in Figure 2-4, each $2\sigma^2$ is adjusted so that the $\mathbf{F}$ matrix off-diagonal column sums are equal to 0.5. This adjustment

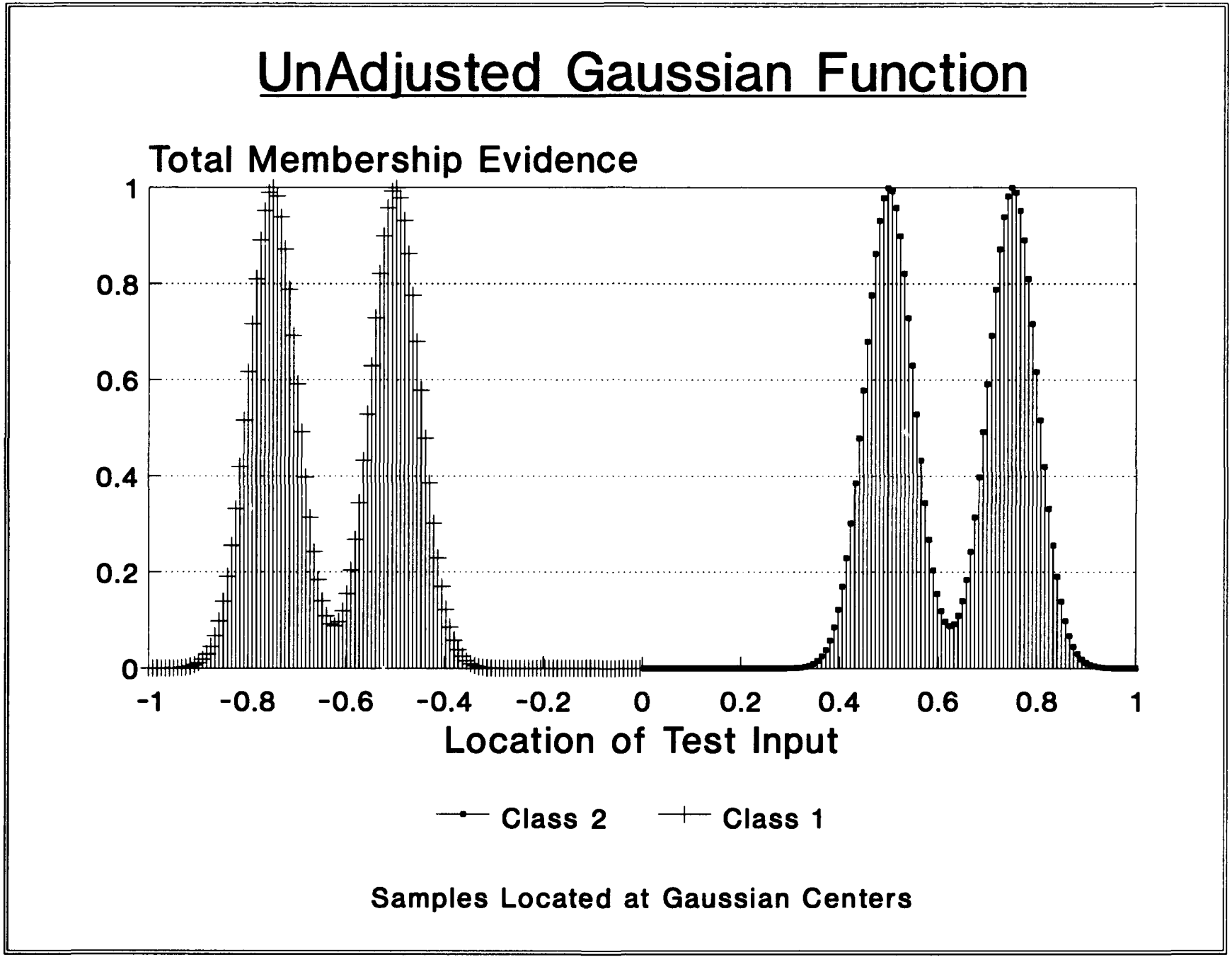

Figure 2-3: Unadjusted Gaussian Function

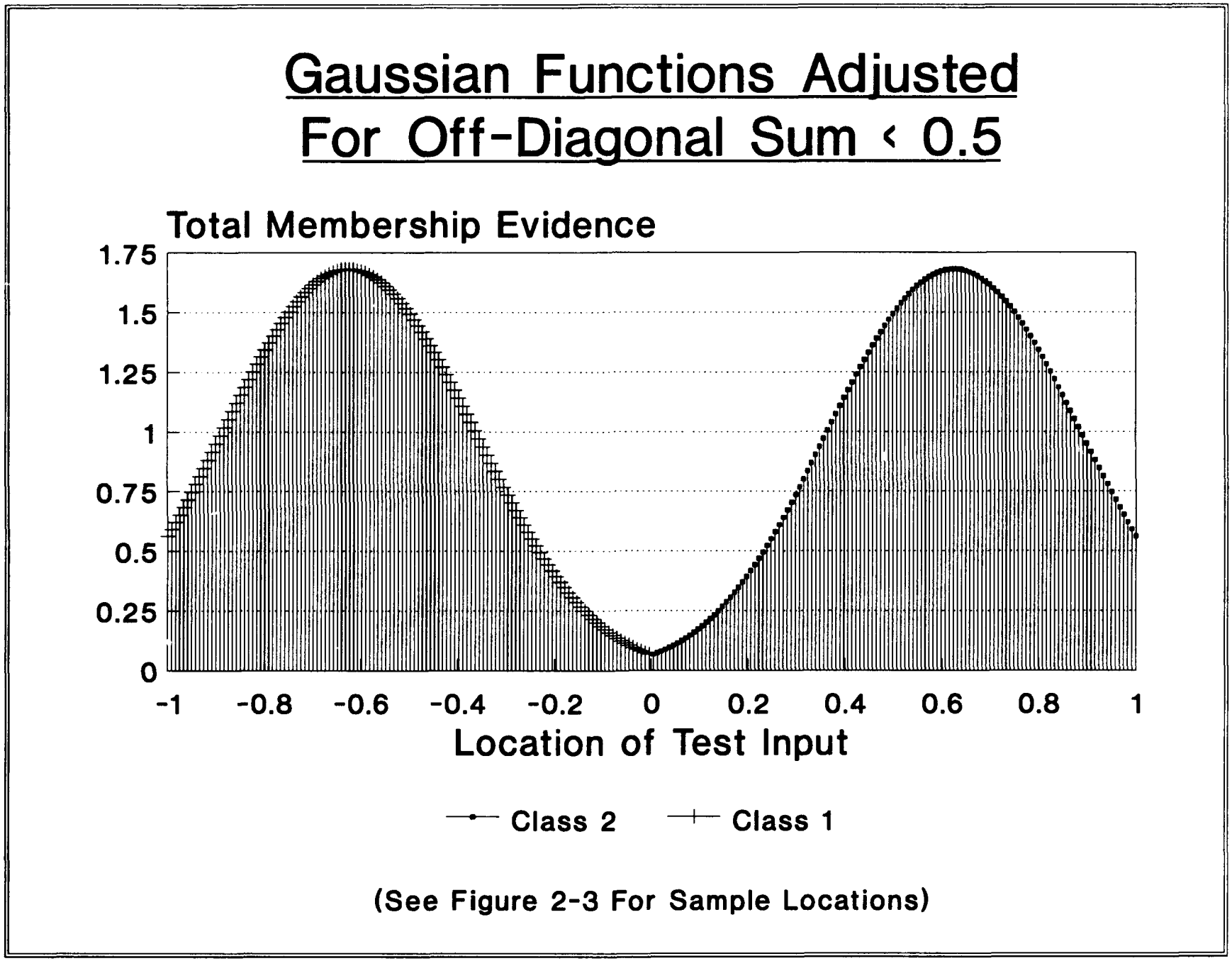

Figure 2-4: Gaussian Functions Adjusted for Off-Diagonal Sum Less Than 0.5

satisfies the columnar diagonal dominance criteria since the diagonal elements are all unity. Figure 2-5 is the same as Figure 2-4 except that the off-diagonal column sum is 0.05. Note that with lower sums generalization decreases and the separation between classes improves.

Interpretation as a Neural Network

The vectors developed during ENN training can be viewed as neural network weights. This section discusses the resulting ENN neural network architecture.

As indicated in Figure 2-6, the ENN architecture connects every node in a lower layer to every node in the layer above. The nodes in a given layer are not connected to other nodes in that layer. Data flows in one direction through the network with the test input at the bottom of the figure and the output at the top. Weight values are placed along the inter-node connection lines. (These weights are explained in subsequent paragraphs.) Each node executes an input and a transfer function. (Rumelhart and McClelland [18, pp46-52] give a detailed discussion on the use of multiple equations in neural network nodes.) The following paragraphs describe the four ENN layers and the equations executed by each layer's nodes.

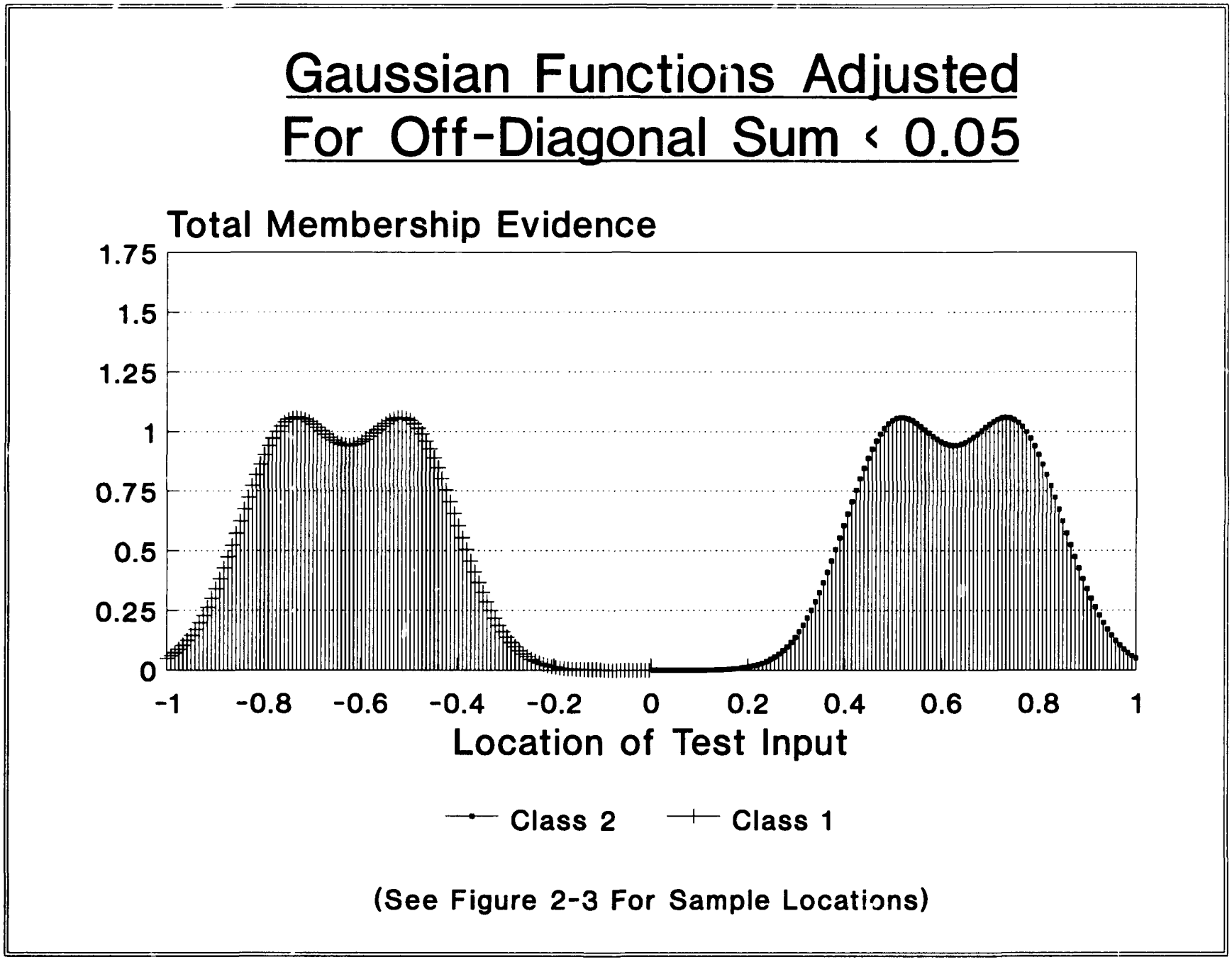

Figure 2-5: Gaussian Functions Adjusted for Off-Diagonal Sum Less Than 0.05

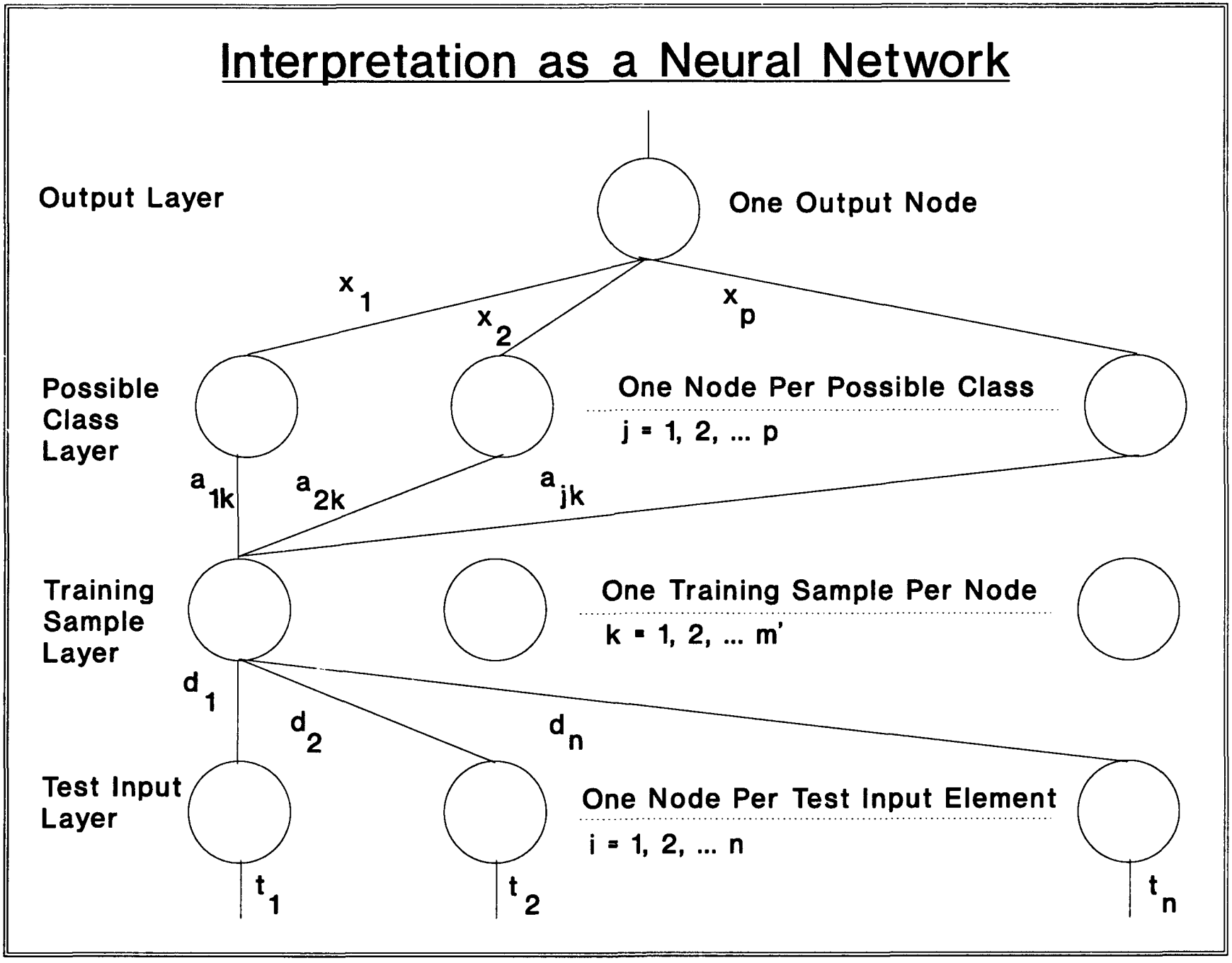

Figure 2-6: Interpretation as a Neural Network

TEST INPUT LAYER: No functions are executed. This layer distributes the test inputs to the layer above.

TRAINING SAMPLE LAYER: Each node in this layer contains a unique member of the entire set of samples. The training data are stored in the nodes so that Figure 2-6 can be simplified.

$$\textit{Input Function} = I_k = \sum_{i=1}^{n} d_i (t_i - s_{ki})^2$$

where d_i is the element discount factor, $0 <= d_i <= 1$,
0 = ignore element, 1 = fully accept element
t_i is a test input element
s_{ki} is an element in sample k
I_k is the input function for k'th sample node

$$\textit{Transfer Function} = c_k = e^{-\frac{I_k}{2\sigma^2_k}}$$

where $2\sigma^2_k$ is calculated according to the columnar diagonal dominance method described earlier. Other methods could be used.
c_k is the voting contribution of the given sample

POSSIBLE CLASS LAYER: Each node in this layer represents a given class or decision alternative.

$$\textit{Input Function} = I_j = \sum_{k=1}^{m'} a_{jk} c_k$$

where a_{jk} associates the k'th sample with the j'th class. While it is interesting to speculate about degrees of association, the authors currently set these weights as follows:

0 = sample k is not associated with class j

1 = sample k is associated with class j

I_j is the input function for class j's node

$$\textit{Transfer Function} = \varepsilon_j = I_j$$

where ε_j is the evidence supporting the selection of class j

OUTPUT LAYER: There is only one node in the output layer. It produces the classification chosen by the network. Alternatively, it could also indicate the ϵ of the selected class. If this layer selects class number zero then it indicates that no class could be selected.

$$Input\ Function = I = h,\ \varepsilon_h,\ and\ \varepsilon_{h-1}\ of$$

$$Max(x_j \varepsilon_j)$$

where I is the input function of the output node
h is the class with the highest ϵ
ϵ_j is the evidence value of class j
ϵ_h is the evidence value of class h
ϵ_{h-1} is the evidence value of the next to highest class
x_j is the expected occurrence of class j. This quantity can also be taken as a risk factor.

$$Transfer\ Function = Selected\ Class =$$

$$h((FOM_h \geq T_F)\ and\ (x_h \varepsilon_h \geq T_\varepsilon))$$

where FOM_h is the figure-of-merit calculated for the class with the highest ϵ, $FOM_h = \epsilon_h / \epsilon_{h-1}$
T_F is the threshold that the FOM must pass before the class can be considered for selection. This means that there must be limited ambiguity in the selection. This threshold should be set to a value that is one or greater.
T_ϵ is the threshold the evidence supporting the class must pass before the class can be considered for selection.
x_h is the expected occurrence of class h
Selected Class is the class chosen by the ENN as the one which best fits the test input

According to Power and Aldag [19, p53] decision makers express significant uncertainty when the probabilities of all alternative

classes are of similar magnitude. Thus, to diminish the uncertainty programmed into the ENN, some attempt should be made to adjust the relative expected occurrence weights connecting the class layer to the output layer. If all these weights are equal, then all classes are expected with equal likelihood. If one class' weight is half that of another class, then the former class is expected to occur half as often as the latter. These weights are largely a matter of judgement. Statistical analysis of the frequency of class occurrences over an extended period of time can aid this judgement. Typically, all these weights can be set to unity unless there is a clear reason to do otherwise. This is consistent with Power and Aldag [19 ,p52] who say that if there is uncertainty about likely outcomes then it is reasonable to assign equal likelihood to all alternatives. Note that uncertainty, in this context, is a situation arising from an inability to assign risk or likelihood factors to alternative decisions [20, p254-255].

Test on 2D Example

Consider the inner circle test illustrated in Figure 2-7. It has two classes whose boundaries are defined by two circles, one inside the other. Class 1 is the inner hub. Class 2 is the outer rim. The filled rectangles are the samples chosen at random from a uniform distribution within each class. Test inputs were chosen in the same way. (The classes are each enclosed by a true circle.

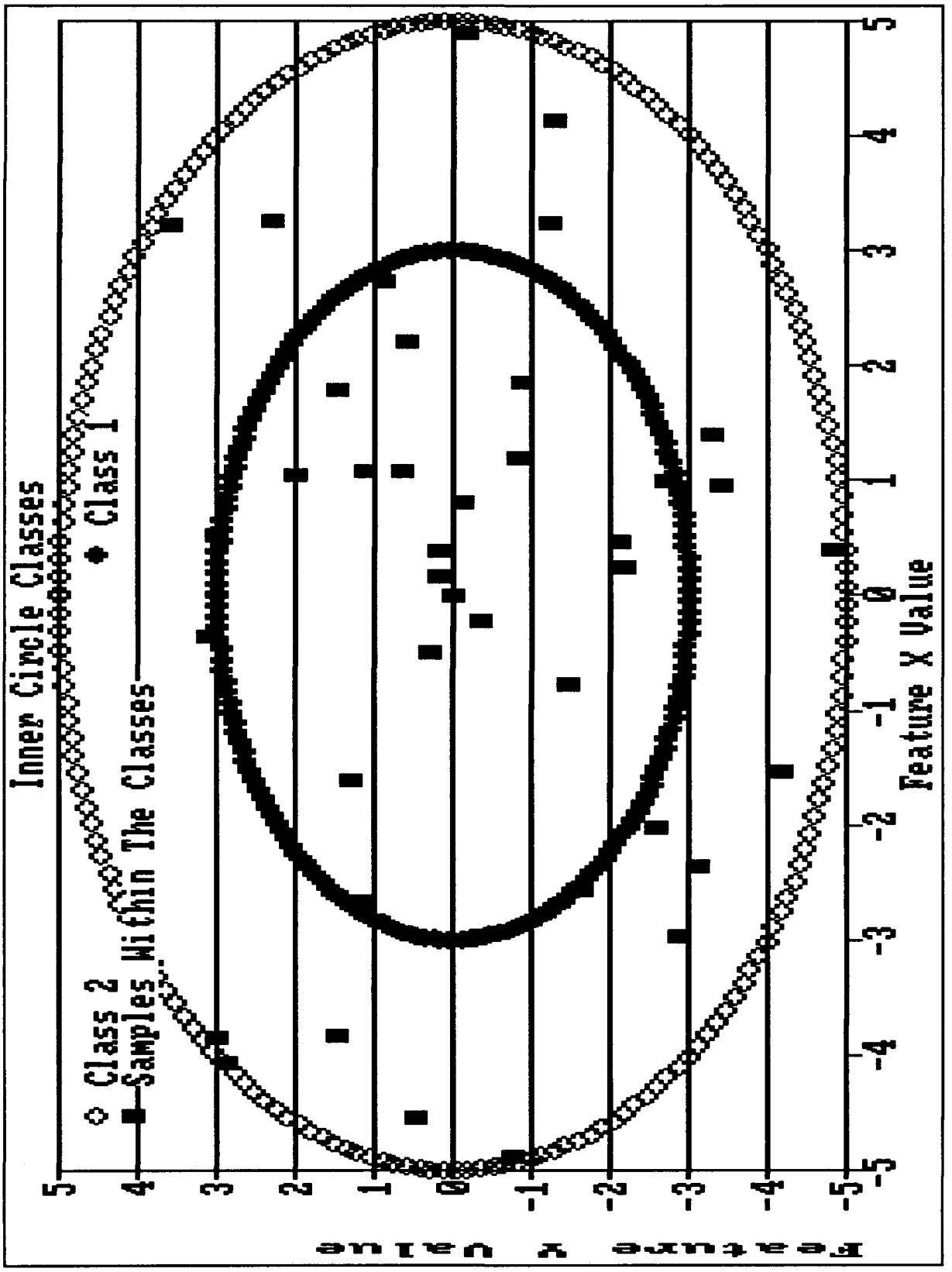

Figure 2-7: Inner Circle Classes

The shape of the graph distorts the image.) Two tests were performed. Their results are reported in Table 1 and summarized in Figure 2-8. Test 2 had twice the number of samples as Test 1. Classifications were based solely on the highest class output from the network. No classification opportunities were denied based on Figure-of-Merit (FOM), evidence, or discount thresholds. The tests were conducted by choosing non-training inputs from within the circles and using the network to determine their respective classes.

Table 2-1. Results of Tests on Inner Circle Domains
(Starting $2\sigma^2$'s adjusted for diagonal dominance goal of 0.05)

Test 1. Starting $2\sigma^2 = 0.005$
Samples Per Class = 10
Test Vectors Per Class = 990

Initial Results:
Class 1 Test Vectors Correctly Classified= 525, % Correct = 53.03
Class 2 Test Vectors Correctly Classified= 214, % Correct = 21.62

Results After Diagonal Dominance Adjustment:
Class 1 Test Vectors Correctly Classified= 583, % Correct = 58.89
Class 2 Test Vectors Correctly Classified= 962, % Correct = 97.17

Test 2. Starting $2\sigma^2 = 0.005$
Samples Per Class = 20
Test Vectors Per Class = 980

Initial Results:
Class 1 Test Vectors Correctly Classified= 681, % Correct = 69.49
Class 2 Test Vectors Correctly Classified= 353, % Correct = 36.02

Results After Diagonal Dominance Adjustment:
Class 1 Test Vectors Correctly Classified= 897, % Correct = 91.53
Class 2 Test Vectors Correctly Classified= 898, % Correct = 91.63

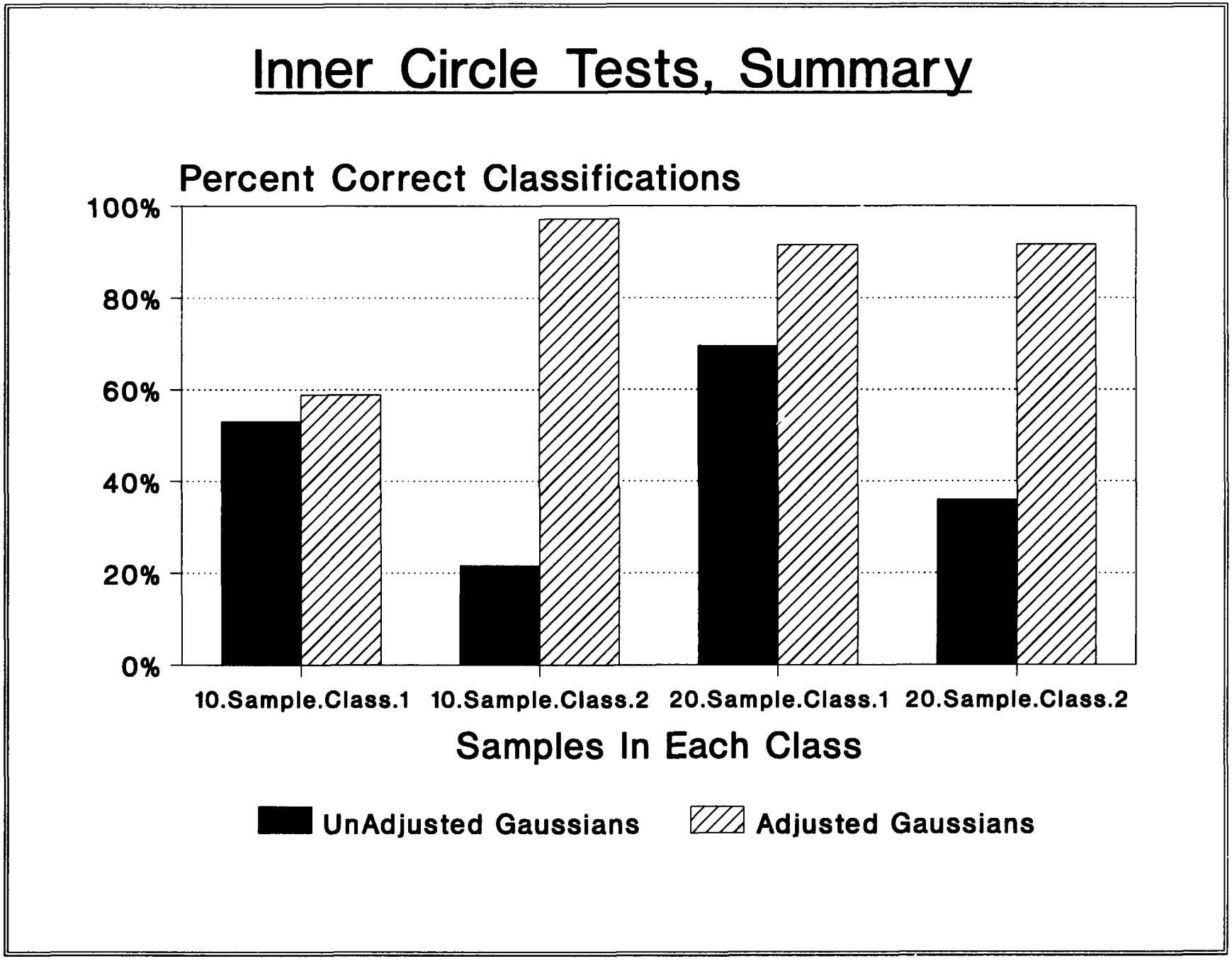

Figure 2-8: Inner Circle Tests, Summary

The overall results improved from Test 1 to Test 2. However, the results for Class 2 were not quite as good in Test 2 as they were in Test 1. This effect is due to the distribution of the additional evidence. Notice that in both tests adjusting the Gaussian functions using the columnar diagonal dominance method caused dramatic improvement in classification accuracy. Clearly, the value of $2\sigma^2$ makes a critical difference in the performance of the network.

Chapter Two Summary

Allowing the class samples to vote for the class membership of a test input is one of many ways to reason under conditions of uncertainty. Neural networks provide an O(1) technology for implementing a voting mechanism.

The voting mechanism implemented by the exponential neural network is based on n-dimensional Gaussian radial basis functions. This network is directly derived from the Stretch and Hammer Neural Network for n-dimensional smooth-surface interpolation proposed by Gustafson et al. [12][14]. This chapter has described an arrangement of that network for classification which is a sub-set of interpolation.

Expert systems and neural networks are two types of knowledge-based systems that seek to make the best decision based on available

information. Their decisions are supported by an accumulation of evidence based on examples of viable alternative decisions or classes. The Gaussian function is a good choice for determining the voting contribution of each example because it is related to the normal distribution which occurs commonly in nature, according to Mendenhall et al. [6, p159]. As the samples accumulate, goodness-of-fit tests can be used to evaluate the fit of other distributions to the examples or to reject the normal distribution.

The training phase of the ENN uses the columnar diagonal dominance method to adjust the Gaussian function variances. This method provides a significant improvement in classification accuracy in the test case tried, compared to using an arbitrary constant variance. The ENN approach is now being subjected to more significant tests that involve the recognition of event sequences in the signal processing arena.

Chapter Two References

1. Gevarter, William B. "The Basic Principles of Expert Systems," article in Expert Systems: A Software Methodology for Modern Applications Los Alamitos, CA: IEEE Computer Society Press, 1990

2. Groothuizen, R.J.P. "Inexact Reasoning in Expert Systems," article in Expert Systems: A Software Methodology for Modern Applications Los Alamitos, CA: IEEE Computer Society Press, 1990

3. Sudkamp, Thomas; M.B. Clausing; and Valerie Cross Machine Intelligence and Threat Identification Systems Cameron Station, VA: National Technical Information Service, Report # WRDC-TR-90-1076, Aug 90

4. Zadeh, Lofti, A. "Fuzzy Logic," article in Expert Systems: A Software Methodology for Modern Applications Los Alamitos, CA: IEEE Computer Society Press, 1990

5. Shafer, G. A Mathematical Theory of Evidence Princeton, NJ: Princeton University Press, 1976

6. Mendenhall, William; Dennis D. Wackerly; and Richard L. Scheaffer Mathematical Statistics with Applications Boston, MA: PWS-Kent, 1990

7. Specht, Donald F. "Probabilistic Neural Networks," Neural Networks, Vol 3, pp 109-118, 1990

8. Ruck, Dennis W.; Steven K. Rogers; Matthew Kabrisky; Mark E. Oxley; and Bruce W. Suter "The Multi-Layer Perceptron as an Approximation to a Bayes Optimal Discriminant Function," IEEE Transactions on Electron Devices, Vol 37, # 12, pp 296-298, Dec 90

9. Blockley, D.I. and J.F. Baldwin "Uncertain Inference in Knowledge-Based Systems," Journal of Engineering Mechanics, Vol 113, # 4, pp 467-481, Apr 87

10. Shepherdson, J.C. "Negation As Failure," Proc: Inference Workshop, ICL, Sep 84

11. Tou, Julius T. and Rafael C. Gonzalez Pattern Recognition Principles Reading, MA: Addison-Wesley, 1974

12. Gustafson, Steven C.; Gordon R. Little; John S. Loomis; and Todd S. Puterbaugh "Stretch and Hammer Algorithm for Data Representation and Generalization," University of Dayton Research Institute paper submitted to Communications in Applied Numerical Methods, Mar 90

13. Specht, Donald F. "A General Regression Neural Network," IEEE Transactions on Neural Networks, Vol 2, # 6, pp 568-576, Nov 91

14. Raeth, Peter G.; Steven C. Gustafson; Gordon R. Little; and Todd S. Puterbaugh Stretch and Hammer Neural Networks For N-Dimensional Data Generalization Cameron Station, VA: National Technical Information Center, Report # WL-TR-91-1146, Oct 91

15. Kabrisky, Matthew and Steven K. Rogers Lectures on Pattern Recognition: Scene Analysis Dayton, OH: Air Force Institute of Technology, 1990

16. Press, William H.; Brian P. Flannery; Saul A. Teukolsky; and William T. Vetterling Numerical Recipes in C Cambridge, England: Cambridge University Press, 1988

17. Zahirniak, Daniel R. Characterization of Radar Signals Using Neural Networks Dayton, OH: Air Force Institute of Technology; Thesis # AFIT/GE/ENG/90D-69; also published by National Technical Information Service (NTIS), Cameron Station, VA; 1990

18. Rumelhart, David E. and James L. McClelland Parallel Distributed Processing, Vol I Cambridge, MA: The MIT Press; 1986

19. Power, Daniel J. and Ramon J. Aldag "Soelberg's Job Search and Choice Model," Academy of Management Review, vol 10, # 1, pp 48-58, 1985

20. Gray, Edmund, R. and Larry R. Smeltzer Management: The Competitive Edge New York, NY: MacMillan Publishing, 1989

(This page intentionally left blank.)

CHAPTER THREE

EXPONENTIAL NEURAL NETWORKS
PRELIMINARY EXPERIMENTS AND OBSERVATIONS

This chapter describes preliminary experiments conducted on the exponential neural network (ENN). Unfortunately, the author's 3-year tour of duty in the Air Force laboratory system came to an end just after these experiments were completed.

Seven types of classification experiments were performed. Experiments 1, 2, and 3 are continuations of the illustration developed in Chapter 2. Experiments 4 and 5 used simulated data while Experiments 6 and 7 used collected data for the application described in Chapter 1.

1. Inner/Outer circle using the ENN

2. Inner/Outer circle using a backpropagation neural network

3. Inner/Outer circle using traditional classification methods (Closest Class Mean and K Nearest Neighbor)

4. Simulated event data incorporating only event-length corruption

5. Simulated event data incorporating both event-length corruption and missed events

6. Collected event data with sub-classes grouped under major classes

7. Collected event data with sub-classes held as separate classes.

The Inner/Outer Circle Experiments

The ENN was tested using the inner/outer circle illustration developed in Chapter 2. To summarize: 2 classes have boundaries which are defined by two circles, one inside the other. Class 1 is the inner hub with a radius of three. Class 2 is the outer rim with a radius of five.

For these experiments, 20 non-duplicative training samples per class were chosen at random from a uniform distribution. One thousand non-duplicative test samples from each class were chosen in the same way. The training set did not have samples which duplicated those in the test set. Each classification method used the training set to determine the respective classes of the test set samples.

The author has concluded that, for this problem, the columnar diagonal dominance method is superior to the fixed-sigma method of training the ENN. A second conclusion is that the ENN results were comparable with backpropagation and K Nearest Neighbor. A better balance of accuracy was achieved by the ENN as compared to backpropagation and K Nearest Neighbor. Closest Class Mean gave results that were clearly inferior to all the other methods.

EXPERIMENT 1: Test results for the ENN are reported in Table 3-1. Classifications were based solely on the highest class output from the network. No classification opportunities were denied based on Figure-of-Merit (FOM), evidence, or discount thresholds.

Table 3-1. Results of Tests with ENN

Columnar Diagonal Dominance Training Method

Goal	% Inner Circle Correct	% Outer Circle Correct
0.05	95	92
0.99	92	95

("Goal" refers to the sum of the Gaussian centered at a given coordinate as calculated at the remaining training samples' coordinates. See Chapter 2 for details.)

Fixed-Sigma Training Method

$2\sigma^2$	% Inner Circle Correct	% Outer Circle Correct
0.0050	62	35
0.6830	94	88
2.7649	93	83
10.0000	100	59
100.0000	100	0

(0.6830 and 2.7649 are the maximum $2\sigma^2$ achieved by columnar diagonal dominance goals of 0.05 and 0.99 respectively.)

From this table, we have concluded that columnar diagonal dominance gives the best overall results of the two methods.

EXPERIMENT 2: Comparison trials were conducted with the same data using a backpropagation network as described by Lippmann. A four-layer model was used, one input layer, two hidden layers, and one output layer. The first hidden layer was connected to the output layer and the input layer was connected to the second hidden layer. A gain of 0.25 and a momentum of 0.0 were used. As with the ENN, the backpropagation network was allowed to learn until it

classified all the training samples correctly. The output layer had two nodes, one for each class. Classifications were based on the same criteria as the ENN. The results of these trials are given in Table 3-2. In no case did backpropagation give a superior balance of accuracy as compared to the ENN.

Table 3-2. Results of Tests With Backpropagation

Backpropagation Training Method

# Nodes	% Inner Circle Correct	% Outer Circle Correct
1	x	x
2	x	x
3	96	79
6	96	88
12	98	76
24	97	83
48	95	88

("# Nodes" refers to the number of nodes in each hidden layer.)

("Balance of results" refers to the fact that, while the inner circle is easy to identify, the outer circle is not. It is desired to not be totally superior with the inner circle to the expense of the outer circle. Both should be above 90%.)

(For # Nodes 1 and 2, backpropagation did not meet the training criteria.)

EXPERIMENT 3: Traditional classification methods were also tried. The training and test data were used to drive a Closest Class Mean and a K Nearest Neighbor (K = 1) analysis. The Closest Class Mean method is designed to give good results if the training samples are well clustered. The K Nearest Neighbor method typically works well

for situations where there is a considerable degree of overlap between classes. Table 3-3 gives the results of this experiment. Neither traditional method's accuracy exceeded that of the ENN.

Table 3-3. Results of Tests With Traditional Methods

Closest Class Mean	
Inner Circle: 63%	Outer Circle: 50%
K Nearest Neighbor (K = 1)	
Inner Circle: 95%	Outer Circle: 86%

The Simulated Event Experiments

These experiments were based on simulations of radar pulse trains. Staggered pulse trains were used. A staggered train has a repeating pattern of pulses. For instance, a repeating pattern of two different pulse lengths might be used, as in (30 40 30 40 ...). (A pulse length is measured from the start of one pulse to the start of the next pulse.) Event-length corruption was achieved by using a pulse train simulator to approximate variations in pulse length because of normal radar operating conditions.

Another method of representing a pulse train is by histogramming. A simple histogram simply sets up a number of bins of gradually

increasing values and increments the counter for each value depending on the pulse lengths found in a given train. For instance, the pulse train (30 40 30 40 30 40) would have the histogram (0 0 3 3) if a four-bin histogram were used with each bin being an increment of 10 (10 20 30 40).

An extended histogram may be formed by successively adding the pulse lengths (according to Moody). A level one sum is performed, then a level two sum, up through a level n sum where n is the number of pulses in the train. For example, the pulse train (30 40 30 40 30 40) would become the following list of event lengths prior to the formation of the extended histogram (30 40 30 40 30 40 70 70 70 70 70 100 110 100 110 140 140 140 170 180 210). If a two-dimensional matrix is set up with the original pulse train elements in row 1 and zeros in row 2, a quick method of calculating the values for binning in an extended histogram is given by the following algorithm:

```
p = # pulses in train
t = pulse train matrix, [2,p]
l = 0
while (l < p)
     k = l
     n = p - l
     for j = 1 to n
          k = k + 1
          t[2,j] = t[2,j] + t[1,k]
          write(disk, t[2,j])
     end
     l = l + 1
end
```

The following progression of rows occurs if the original histogram of (30 40 30 40 30 40) is used by this algorithm:

Columns -->	1	2	3	4	5	6
Row 1	30	40	30	40	30	40
Subsequent Row 2s	0	0	0	0	0	0
	30	40	30	40	30	40
	70	70	70	70	70	
	100	110	100	110		
	140	140	140			
	170	180				
	210					

Once the extended pulse train is formed, a histogram is accumulated. If the above train were used to form a histogram that had bins incrementing by 50, the following bins would result (50 100 150 200 250). This would cause a histogram of (6 7 5 2 1). Clearly, the number of bins in a histogram has a dramatic affect on the accumulation in each bin and, thus, in the feature values fed to the neural network for classification.

Of course, there are many other ways to represent a pulse train. It is important to note that input vector representation; feature extraction; and data sensing, capture, transmittal, storage, and management are the most involved and time consuming activities in neural network implementation. Most researchers concentrate on the intellectually challenging parts but tend to forget such activities involving the data, assuming them away. It has been this author's experience that while these two activities may seem technically trivial, they are usually what stands in the way of a successful implementation (if one ignores user acceptance).

In these experiments, the repeating pulse patterns listed in Table 3-4 were used. The pulse train simulator did not produce only one corrupted representation of each normal pattern. However, the variation was only in the decimal places. Each pulse train contained 12 pulses. There was one uncorrupted pulse train per class for training (eight total) and eight corrupted pulse trains per class for testing (64 total). The task of the ENN was to learn the training samples and correctly classify the test samples.

Table 3-4. Simulated Training and Test Patterns

Class	Normal Pattern (Training)				Corrupted Pattern (Testing)			
1	297.8				293.3			
2	100.0				93.3			
3	297.8	344.3			291.9	336.8		
4	100.0	200.0			89.8	190.8		
5	297.8	344.3	200.0		293.3	340.0	200.0	
6	100.0	200.0	300.0		100.0	200.0	300.0	
7	297.8	344.3	200.0	100.0	291.8	336.7	190.8	89.8
8	100.0	200.0	300.0	400.0	89.8	190.8	291.8	392.8

The experiments showed that all training methods achieved 100% correct classification from zero through two missed events. Up through four missed events, there was opportunity for perfect performance depending on the training method used.

EXPERIMENT 4: The ENN was tested using simulated event data incorporating only event-length corruption. For both the simulated and collected event experiments, three methods were used for training:

1) Columnar Diagonal Dominance (Described in Chapter Two. Training "Goal" refers to the desired sum of the Gaussian centered at a given coordinate as calculated at the remaining

training samples' coordinates.)

2) Mid-Position Sigma (Training "Goal" refers to the desired value of the Gaussian centered at a given training sample's coordinates calculated at the half-way point between the sample and the nearest neighbor. The method for calculating Sigma under this requirement is described below.)

3) Mid-Position Maximum Sigma (All Sigmas are set to the largest value generated during Mid-Position Sigma training.

The value of Sigma in the Mid-Position Sigma training method is set in the following way. Assume that a training sample is located at coordinates held in vector **X** and that its nearest neighbor's coordinates are in vector **N**. Recall that the Gaussian equation which specifies g, the value at **N** of the Gaussian centered at **X** is:

$$g = a\, e^{\frac{-\sum_{i=1}^{k} (x_i - n_i)^2}{2\sigma^2}}$$

where:

σ^2 is the variance of the Gaussian function centered at **X**
e is the base of natural logarithms
k is the number of elements (features or dimensions) in **X** and **N**
x_i is the i'th element in vector **X**
n_i is the i'th element in vector **N**
g is the goal for the Gaussian result
a is the desired peak of the Gaussian at **X**
(In these experiments, this variable was set to 1.0)

The summation is simply the square of the distance between **X** and **N**. To compute the Gaussian value at some intermediate point between **X** and **N** it is necessary to take the square root of this summation; multiply by f, the fraction of the distance desired between **X** and **N**; and then square the result. This equation is:

where:

d is the distance to the intermediate point squared

$$d = \left(f \sqrt{\sum_{i=1}^{k} (x_i - n_i)^2} \right)^2 = f^2 \sum_{i=1}^{k} (x_i - n_i)^2$$

f is the fraction of the distance between **X** and **N**
(In these experiments, this variable was set to 0.5)

Having computed a value for d and set a goal value for g, an algebraic rearrangement of the Gaussian equation gives the necessary value for Sigma.

$$g = a\, e^{-\frac{d}{2\sigma^2}}$$

becomes

$$2\sigma^2 = -\frac{d}{\ln\left(\frac{g}{a}\right)}$$

which can be rewritten

$$\sigma = \sqrt{-\frac{d}{2 \ln\left(\frac{g}{a}\right)}}$$

It is assumed that the value of a is greater than the value of g.

For all three training methods, testing was conducted with goals of 0.003125 and 0.9. (The smallest value possible before a computer computational error occurred was 0.003125.) In all attempts, the ENN correctly classified 100% of the test samples.

EXPERIMENT 5: The ENN was tested using simulated event data which contained both event-length corruption and missed events. In the case of a missed event, the length of the current event is extended until the next detected event occurs. For example: Consider the event train (30 40 50 60 70). If one event is missed, this train becomes (50 60 70 70 70). If two events are missed, the original train becomes (70 110 70 70 70). As events are "missed" in the original train, the length of the missed event is added to the previous event. Then, the train is shifted left to remove the missed event. The last event length value stays in the right-most position. All event trains are rotated left until the lowest event length is in the left-most position. It may be argued that missed events can occur at random times in the event sequence and that a missed event might cause the event sequence to have fewer elements. Thus, there are many ways to model event sequences that have missing events.

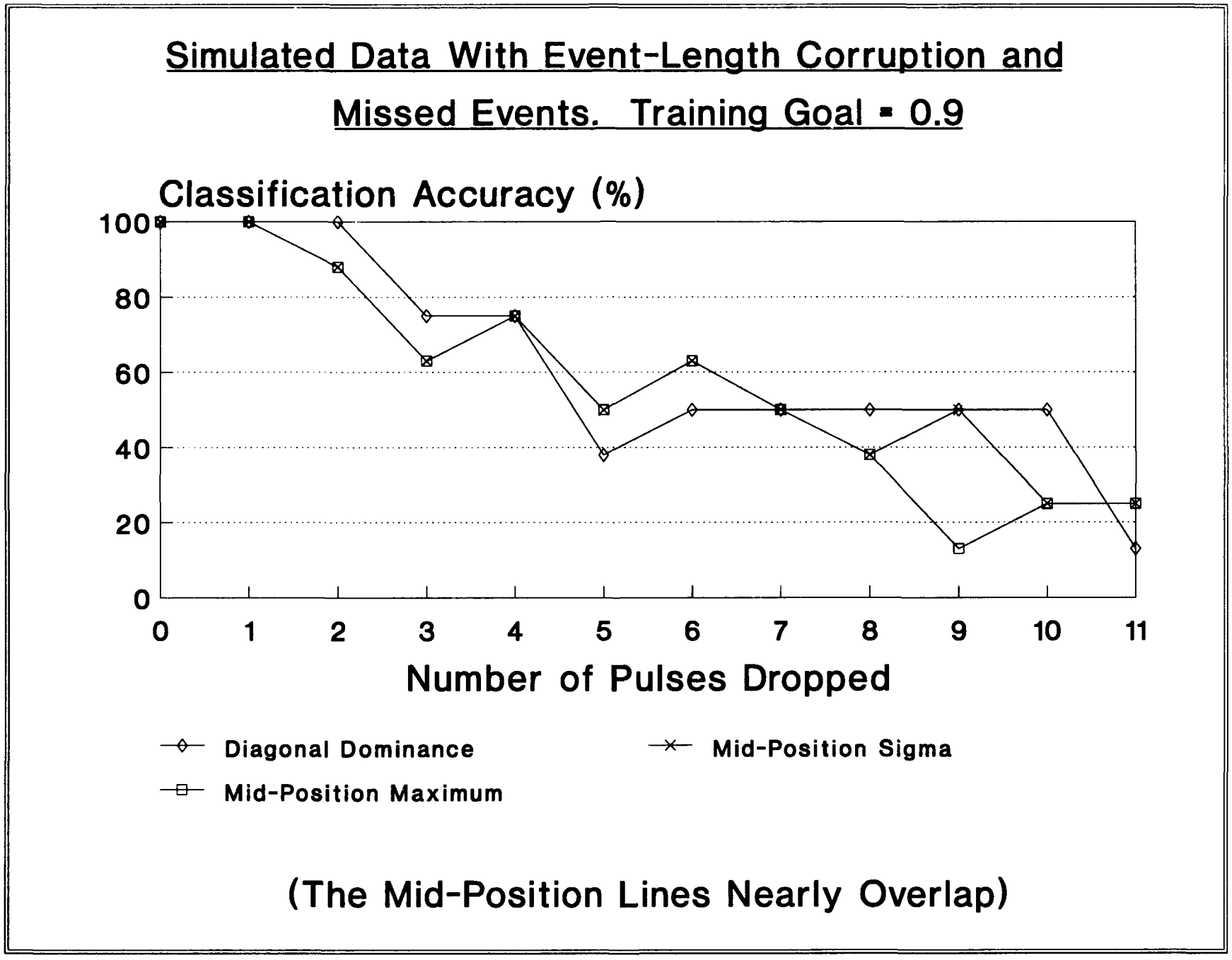

Figure 3-1: Simulated Data With Event-Length Corruption and Missed Events. Training Goal = 0.9

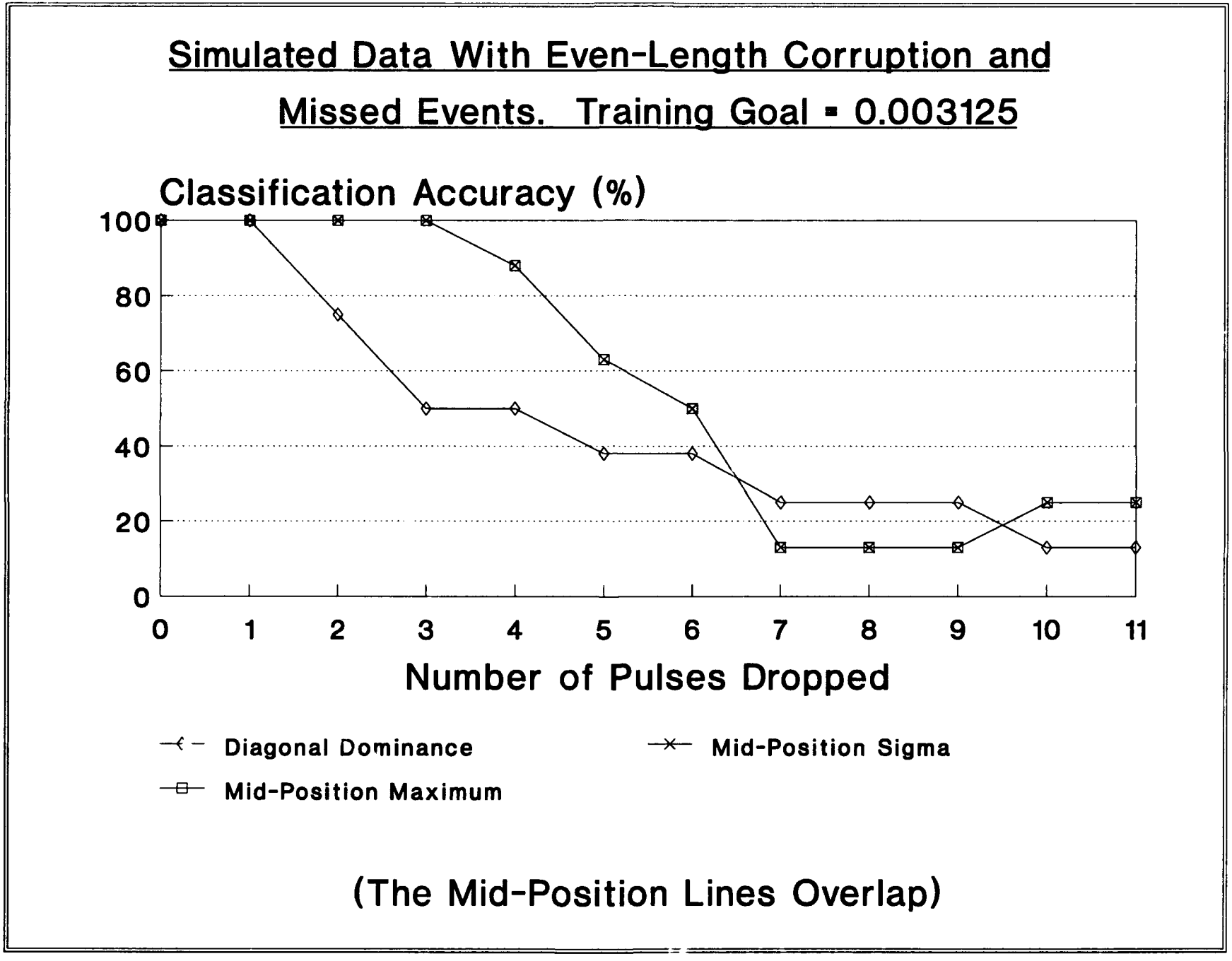

Figure 3-2: Simulated Data With Event-Length Corruption and Missed Events. Training Goal = 0.003125

Figures 3-1 and 3-2 show the classification accuracy of the ENN under the conditions of this experiment. The ENN was allowed to always produce a classification. In Figure 3-1 a training goal of 0.9 was used. For this case, the results of all three training methods was generally comparable, with columnar diagonal dominance faring somewhat better than the others, keeping classification accuracy through three missed events. In Figure 3-2 a training goal of 0.003125 was used. In this case, columnar diagonal dominance did not fare as well, losing classification accuracy after two missed events.

It is possible to set the output threshold so high that the ENN only makes correct classifications. However, relatively few test samples will actually be classified. The rest will simply fall into an "unclassified" category. The ENN will, thus, not classify certain test samples whether or not it could have done so correctly. Naturally, the number of test samples not classified increases as the number of missed events increases. Figures 3-3 and 3-4 show the percentage of classified test samples when the previous trials were rerun with the output threshold set so high that no classification errors were made. Note that each graphed point represents 100% correct classifications when a classification was attempted. This may be compared to Figures 3-1 and 3-2 which show the percentage correct classification when all test samples were classified. The comparison gives an idea of the number of missed opportunities

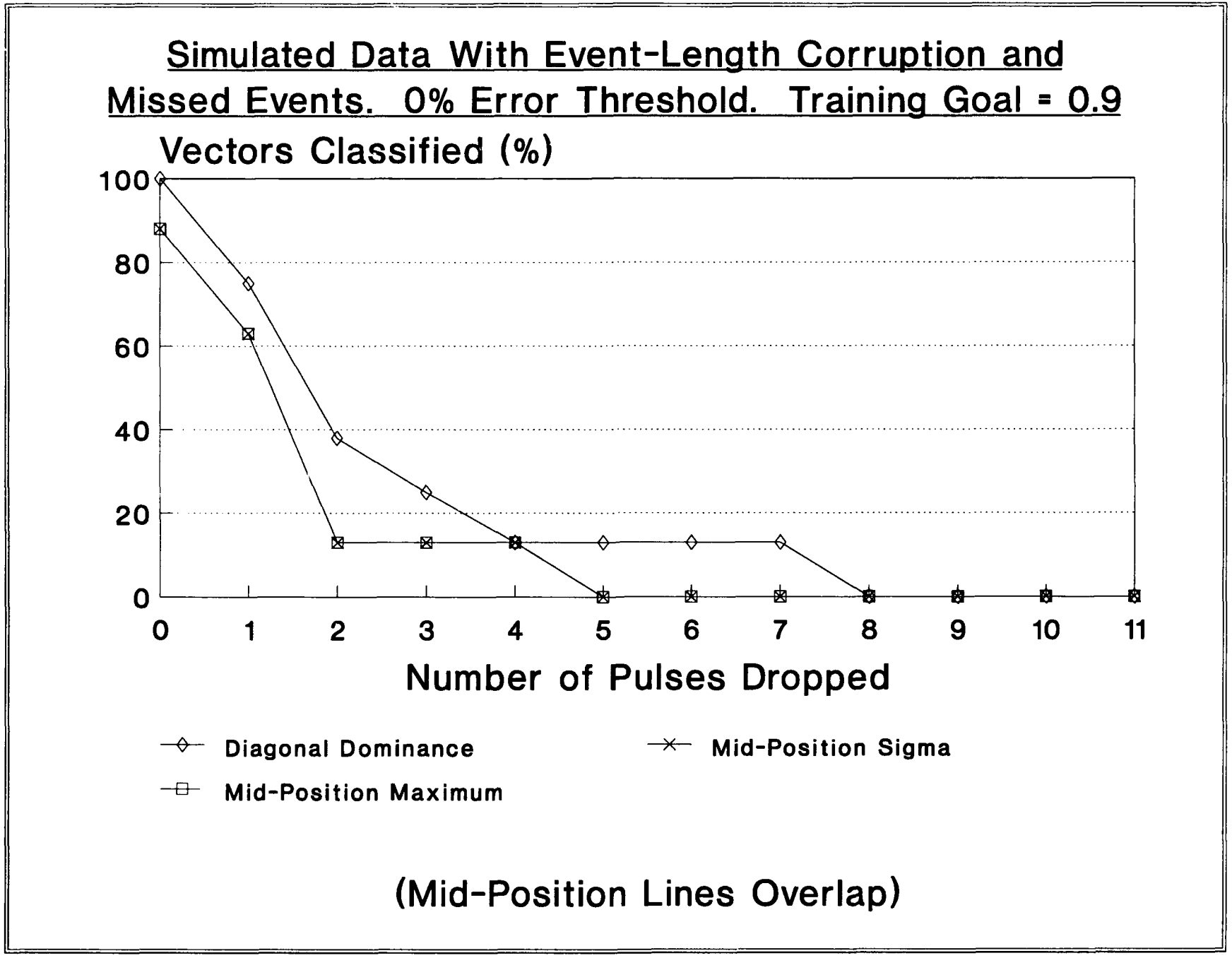

Figure 3-3: Simulated Data With Event-Length Corruption and Missed Events. 0% Error Threshold. Training Goal = 0.9

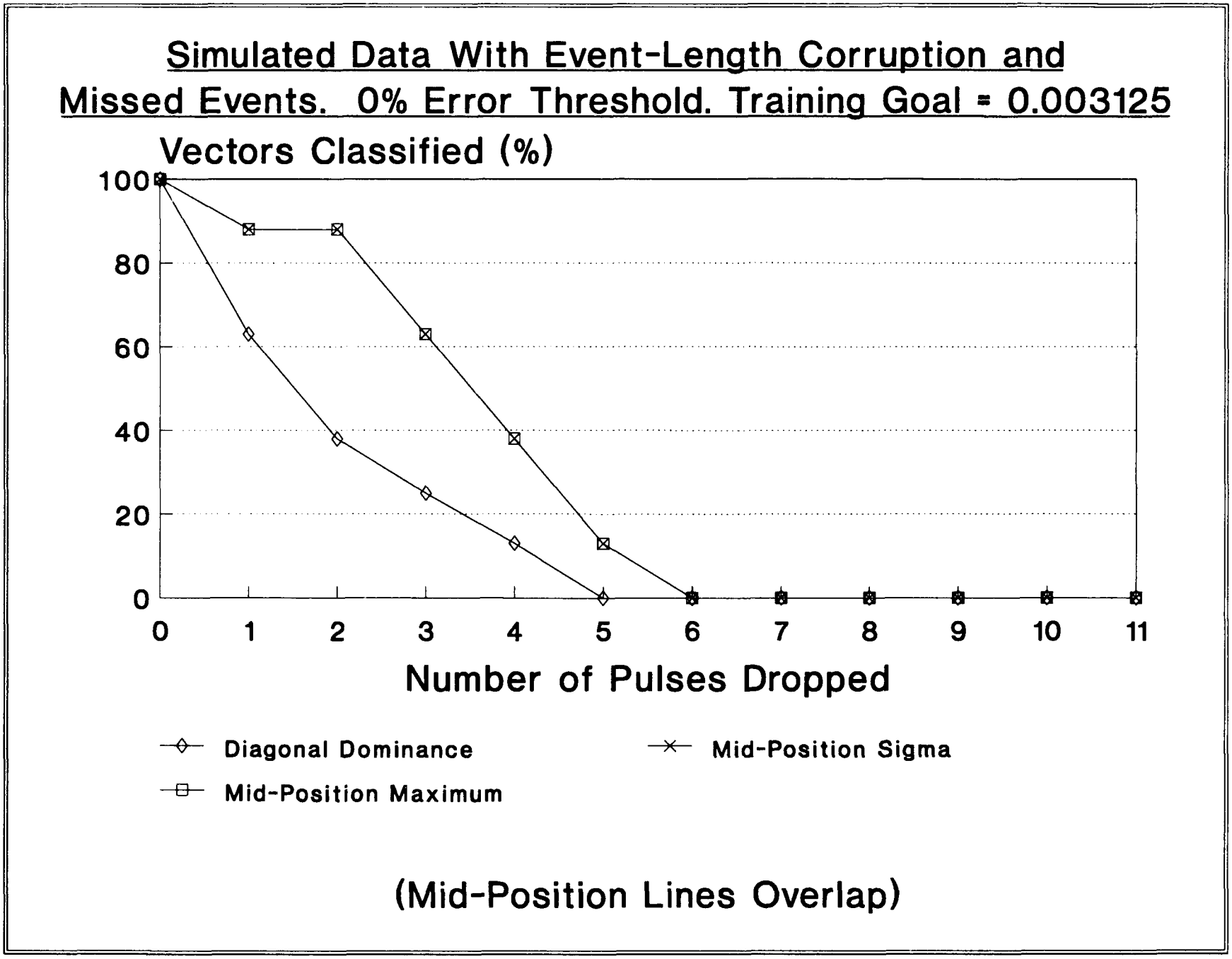

Figure 3-4: Simulated Data With Event-Length Corruption and Missed Events. 0% Error Threshold. Training Goal = 0.003125

for a correct classification. The output threshold is generally set according to judgements on the cost of error. In this experiment, it was assumed that a 0% error rate was desired. For this part of the experiment, diagonal dominance performed better with a large training goal. The mid-position methods did better with small training goals.

The Collected Event Experiments

Tests were conducted using pulse trains collected via antenna from two different radar systems. These systems operated in various modes. The data were collected in such a way that it was possible to associate the pulse train with a radar system and its operating mode. Thus, it was possible to group the data under a major class (a given radar system) or under sub-classes (a given radar system operating in a given mode). The experiments compared training goal to classification accuracy. The pulse trains contained 24 pulses each.

In these experiments, the Mid-Position Maximum Sigma training method performed best if the training goal did not exceed 0.60. In the case of training goals between 0.80 and 0.90, the Diagonal Dominance method was better. In only one case did Mid-Position Sigma offer the best results. It might generally be inferred from the collected event experiments that consideration of the position of the training samples did not always offer a chance to improve

classification accuracy.

EXPERIMENT 6: In this experiment, the operating modes were grouped under their respective radar. The resulting two-class data was used to train the network. There were 244 training samples for the first radar system and 671 for the other. 427 test samples per radar were presented to the trained network.

Figure 3-5 shows the results of this experiment. The network was trained using several training goals. The Mid-Position Maximum Sigma training method gave consistently superior results until a large training goal was used (0.6). At this point, the network tended to over-generalize and the performance dropped off significantly. Diagonal Dominance gave the most consistent results but never achieved more than 71% accuracy, whereas Mid-Position Maximum Sigma achieved 87% over a fairly large range of training goals (0.003125 through 0.50).

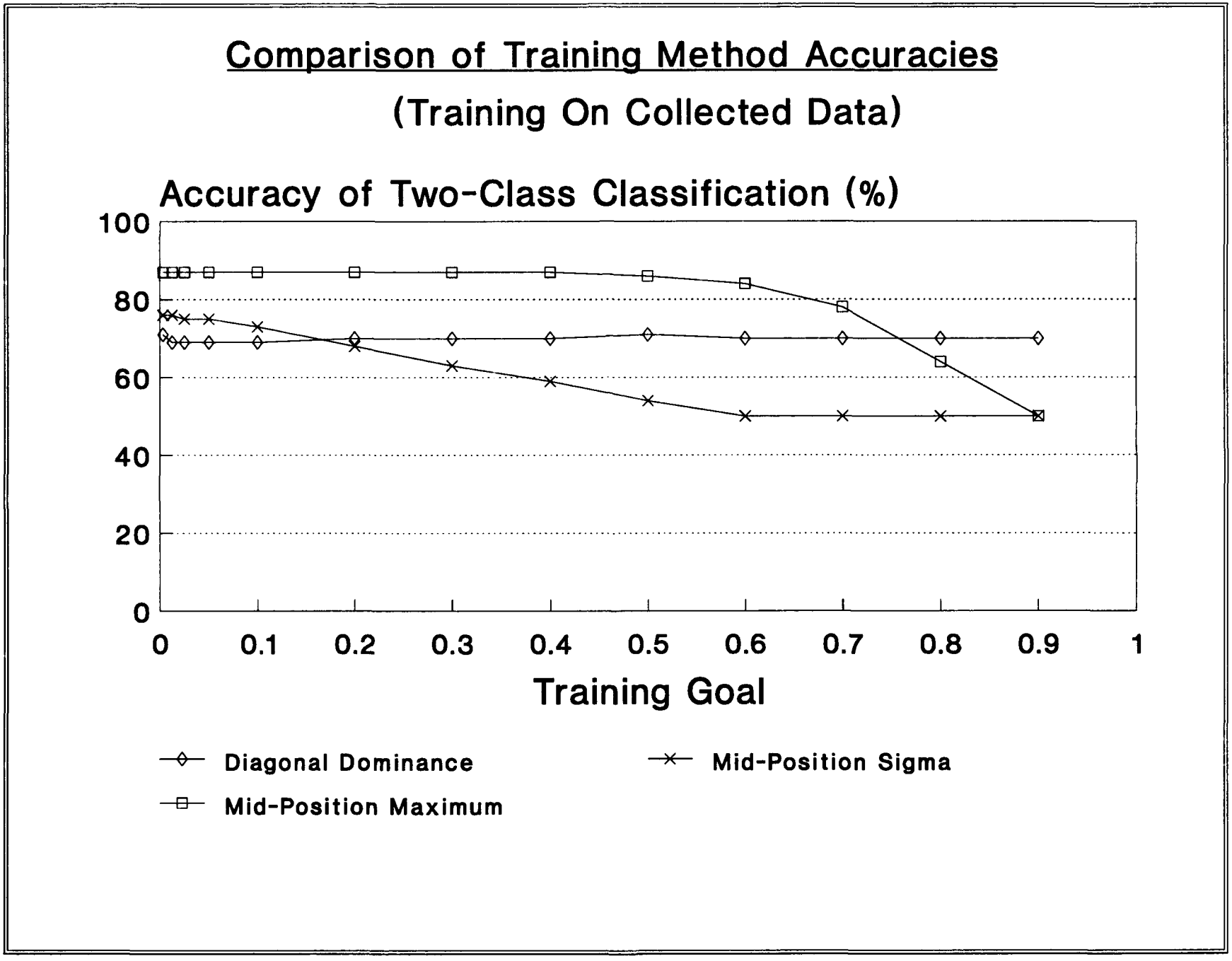

Figure 3-5: Comparison of Training Method Accuracies Using Collected Data in the 2-Class Problem

EXPERIMENT 7: In this experiment, the training data from the two radar systems were split into 14 classes representing the same number of radar operating modes. Each class had 61 training samples and 61 test samples. Figure 3-6 shows the results of this experiment.

Note that the best results were achieved with the highest training goal. This may be explained by the low volume of training data. The high training goal permitted maximum generalization, which tended to compensate for the low volume of training samples. However, accuracy above 59% was not achieved. The low volume of training samples was simply not sufficient, in spite of the high degree of generalization. The low training data volume and the generally poor results are reasons for not trying to infer too much from this experiment. It appears clear that a larger training data volume is necessary.

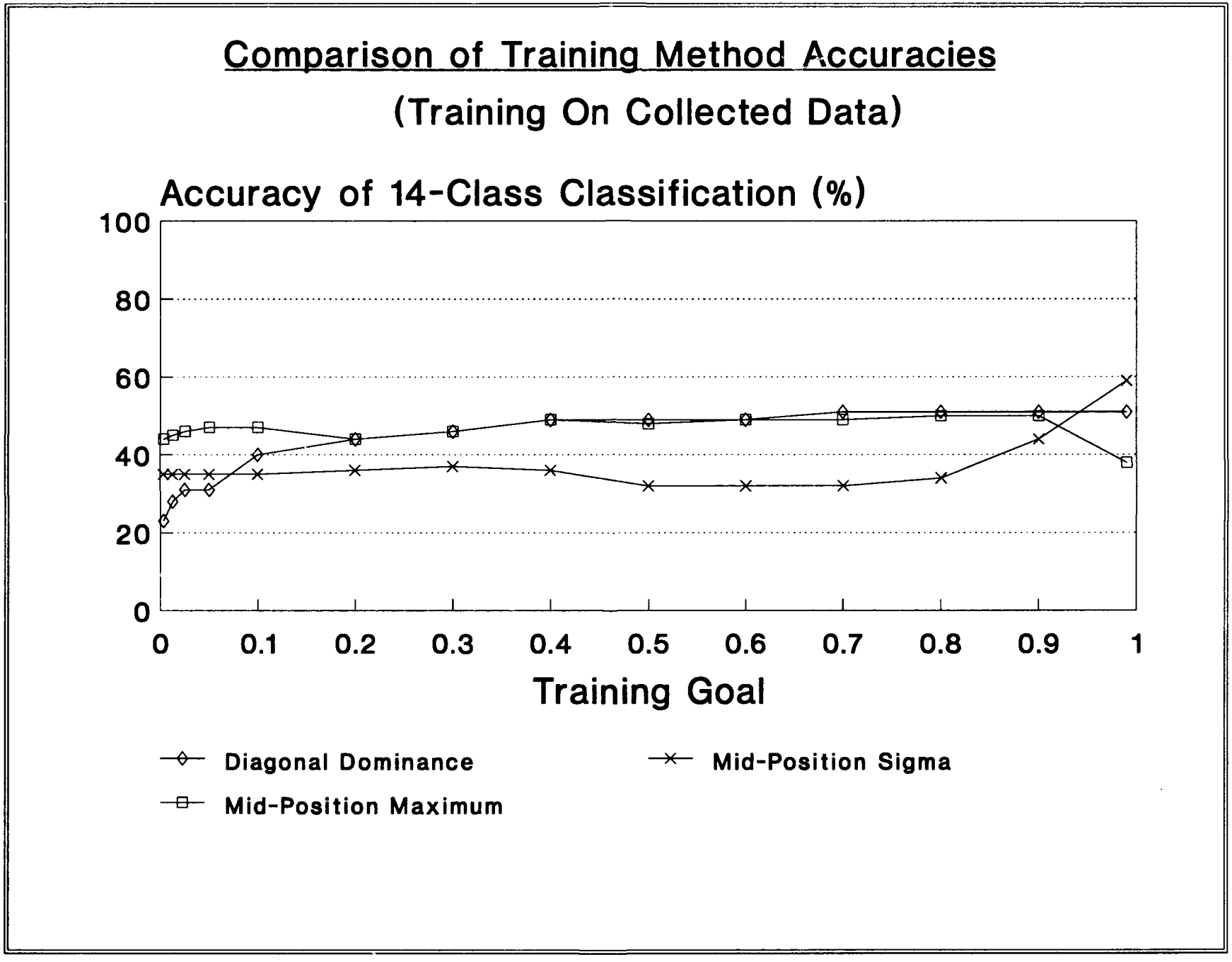

Figure 3-6: Comparison of Training Methods Using Collected Data in the 14-Class Problem

Suggested Additional Experiments

Orders have arrived so it is time to move on. However, some suggestions should be made if work is to continue in the future.

As an interesting side trip, some extremely wide fixed Gaussians were tried with Experiment 7's data. Results up to 65% accuracy were obtained. An experiment could be attempted on an increased training data volume for the 14 radar modes which simply fixes the width of all Gaussians at the same value. A series of values could be tried, from very small, to very large. In this way, the effects of data volume combined with Gaussian width might be observed for training based on collected data. In fact, a three-dimensional graph could be prepared, showing on one axis increasing data volume. The other axis would show increasing Gaussian width. This would be a valuable experiment since Experiments 6 and 7 tended to show that consideration of the position of all training samples was not always effective.

These experiments attempted to match unknown pulse trains with the closest train in the training set. A metric of accuracy needs to be developed that gives a degree of error, instead of just right or wrong. An incorrectly matched train might well be close enough. It needs to be determined which is better, to use the corrupted train or the closest known train, in subsequent processing. One is tempted to use a simple distance metric. The distance of the

corrupted train from the actual train would be compared to the distance of the matched train from the actual train. The train with the least distance would be termed "better." However, the application would have to lend some information to the choice of metric.

A statistical measure of class overlap, as reflected in the training data should be developed. This would give an indication of the separability of the classes and thus an expectation of classification accuracy. Fossum and Weingard have used such a metric successfully. They calculate the class overlap and then assume a uniform distribution of the test data within the classes. From this, they determine how much of the test data would fall into only one class. They have indicated that the assumption of uniform distribution can be removed but only at a very high computational cost. Their technique is based on standard statistical methods and provides an excellent criteria which supports a metric for evaluating neural networks. This metric compares the performance of the neural network to the Bayes theoretic performance.

An attempt should be made to test the network with collected data and have the training conducted with the actual pulse train generated by each radar system mode. In this way, one vector per mode would be used for training. The training and testing in Experiments 5 and 6 were conducted in this way, albeit with simulated test data. In all of the pulse train experiments, which

involved some form of data corruption, Diagonal Dominance gave better results for large training goals. The Mid-Position methods tended to work better for small training goals. The Inner/Outer Circle experiments did not involve any data corruption. In those experiments, the Diagonal Dominance method worked best for high and low training goals. Thus, it would appear as if the type of training data affects the performance of the different training methods.

Raeth et al. described their successful use of a training data "stretching" technique which tends to mitigate a problem's indeterminacy. This technique could be tried here as well. The pulse lengths are simply taken as discrete features without regard to their specific sequence so the technique should apply. However, some exploration of "stretching" the pulse train before or after histogramming would have to be performed.

Comments on the ENN

Schlag has indicated that her experiments show that, for the data she tried, the classification accuracy of networks trained via backpropagation is comparable to that of traditional methods. The Inner/Outer Circle experiments reported here show comparable accuracy between the ENN, backpropagation, and K Nearest Neighbor, with the ENN being superior overall. The argument of comparability with iteratively trained networks and traditional methods is

strengthened by Ruck's proof that iteratively trained networks approximate Bayesian optimal discriminators. Specht has shown that Gaussian basis function networks can approximate PDFs (the ENN is related in many ways to Specht's work). That those who compare neural networks to traditional methods find that comparable accuracy seems normal when one speculates on their common mathematical bases.

The one issue with the ENN is that the number of network nodes, in the hidden layer, increases linearly with the number of training examples. Some traditional classification methods also have this problem in that they need linearly increasing memory as the number of examples increases. Execution time can also increase with traditional methods because of their non-parallel nature.

The issue of linearly increasing network size is alleviated by two factors: 1) advances in the hardware available for implementing massive arrays of parallel processors and 2) methods for limiting the number of samples required for representing a given problem.

Given the rapid progress of current work in multi-processor chip development and the off-shelf availability of transputers, it is safe to say that network size is not an issue for many realistic problems involving massive arrays of parallel processors. As the research continues, future problems will be accommodated as well. Transputers (from Inmos) offer high-speed, course-grain (many

network nodes per processor), parallelism in a small space. They are well into the stage of commercial viability. Both Intel and MicroDevices have placed neural network prototype chips on the market. These are the beginnings of fine-grain (one network node per processor) parallelism with many processors per chip. Indeed, Agranat et al. are conducting research that has led to the design of a chip which contains 256 fully interconnected processor-nodes. Kampf et al. have developed a way to minimize the number of interconnecting "wires" needed when large numbers of processor-nodes are placed in one network. An overview of network multi-processor hardware is given by Collins and Penz. These trends in chip development indicate that the decrease in processor size and thus the number of processors possible per chip tends to negate the issue of a linear dependence of network size on the number of samples.

Burrascano has developed a method for limiting the number of samples required in a Gaussian basis function network. He uses a vector quantization technique which develops reference samples in dense clusters without overly affecting the accuracy of the network.

The issue of linear dependence of network size on the number of samples is addressed by advances in hardware and data analysis techniques. Therefore, it can be asserted that the ENN has much in the way of practical value in classification problems for the

following reasons:

a) bounded resource levels for training and execution

b) fixed execution time regardless of problem complexity, when implemented on fine-grain parallel hardware, and a minimized growth in execution time when executed on course-grain parallel hardware

c) traditional classification techniques are not always as obviously implementable in a parallel processing fashion

d) parallel implementations of statistical theory exist but they have the disadvantage of more complex time- and memory-consuming equations

e) the ENN offers a straight-forward means of programming a massive array of parallel processors to support classification decisions (As such, it is not necessary to attempt to determine the natural parallelism in traditional classification algorithms or to "coax" an iterative network to train.)

f) The linear relationship between network size and the number of training samples is not a problem due to modern hardware technology and methods to limit the number of training samples required.

g) By permitting the nodes in some layers to execute a somewhat more complex equation than is usual in traditional neural networks, the network architecture is simplified. The use of complex nodes is non-traditional but not without precedent as seen in the works of Morgan et al. and Klopf.

Chapter 3 References

Agranat, A.J.; C.F. Neugebauer; R.D. Nelson; A. Yariv; "CCD Neural Processor: A Neural Network Integrated Circuit with 65536 Programmable Analog Synapses," IEEE Transactions on Circuits and Systems, vol 37, # 8, pp 1073-1075, 8 Aug 90

Burrascano, Pietro, "Learning Vector Quantization for the Probabilistic Neural Network," IEEE Transactions on Neural Networks, vol 2, # 4, Jul 91

Collins, Dean R. and Andrew P. Penz, "Considerations for Neural Network Hardware Implementations," Proc: IEEE International Symposium on Circuits and Systems, vol 2, 1989

Fossum, Jeff O. and Fred S. Weingard; personal communication; Booz-Allen & Hamilton, Artificial Intelligence Practice, Mar 92

Kampf, F.; P. Koch; K. Roy; M. Sullivan; Z. Delalic; S. Dasgupta, "Digital Implementation of a Neural Network," Proc IJCNN International Joint Conference on Neural Networks, Jun 89

Klopf, A. Harry and James S. Morgan, "The Role of Time in Natural Intelligence: Implications of Classical and Instrumental Conditioning for Neuronal and Neural-Network Modeling," chapter in Learning and Computational Neuroscience: Foundations of Adaptive Networks; Cambridge, MA: MIT Press; 1990

Klopf, A. Harry; "A Neuronal Model of Classical Conditioning," Psychobiology, pp 85-125, Vol 16 # 2, 1988

Klopf, Adam Harry; Design and Simulation of Locally and Globally Adaptive Multilevel Networks, dissertation, University of Illinois Medical Center; available from University Microfilms, Ann Arbor, MI; 1971

Lippmann, Richard P. "An Introduction to Computing with Neural Nets," IEEE ASSP Magazine, Apr 87

Moody, Douglas E., personal communication, Mercer University, Macon, GA 1990

Morgan, James S.; Elizabeth C. Patterson; and A. Harry Klopf, "Drive-Reinforcement Learning: A Self-Supervised Model for Adaptive Control," Network, pp 439-448, Vol 1, 1990

Raeth, Peter G.; Steven C. Gustafson; Gordon R. Little; and Todd S. Puterbaugh Stretch and Hammer Neural Networks For N-Dimensional Data Generalization Cameron Station, VA: National Technical Information Center, Report # WL-TR-91-1146, Oct 91

Ruck, Dennis W.; Steven K. Rogers; Matthew Kabrisky; Mark E. Oxley; and Bruce W. Suter "The Multi-Layer Perceptron as an Approximation to a Bayes Optimal Discriminant Function," IEEE Transactions on Electron Devices, Vol 37, # 12, pp 296-298, Dec 90

Schlag, Katharine L.; personal communication; Georgia Tech Research Institute, Systems Engineering Laboratory; Feb 92

Specht, Donald F. "Probabilistic Neural Networks," Neural Networks, Vol 3, pp 109-118, 1990

APPENDIX A

PULSE-TRAIN RECOGNITION AND RESTORATION PROBLEM STATEMENT

When a radio frequency (RF) signal is deinterleaved, the pulses are assigned to various pulse trains according to the deinterleaver's algorithm. For instance, the deinterleaver may decide that two separate pulse trains are contained in the received RF signal. The deinterleaver will then assign the pulses it sees to one of the two trains according to the deinterleaving algorithm. The number of trains perceived to be present is not a constant.

Since the receiver system and the deinterleaving process are not perfect, it can happen that any given pulse train will have wrongfully added pulses, missing pulses, and/or pulses of the wrong length. It would be useful if the pulse trains could be restored to their original broadcast form. A restoration of pulse trains would support deinterleaving and subsequent processing on the pulse train itself and processes which use the pulse train as input.

We can assume that deinterleaved pulse trains contain useful accuracy and that they can be compared to expected pulse trains. Then, the closest known pulse train (within some margin) could be used to replace the unknown train. Pulses from the unknown train that are not used in the new train could be sent back to the incoming pulse buffer for reassignment.

(This page intentionally left blank.)

APPENDIX B

A BRIEF OVERVIEW OF NEURAL NETWORKS

From an applications point of view, the field of neural networks investigates ways to program massive arrays of parallel processors to perform useful tasks. The interest in this field stems from the difficulty of using traditional sequential software methods to meet modern requirements. Another impetus to the field is that the traditional uniprocessor architectures are increasingly the cause of bottlenecks that prevent timely task completion, especially in real-time environments. Finally, neural networks can generalize from examples composed of multiple data elements.

Neural networks are massive arrays of simple processors that execute in parallel. These processors are typically arranged in layers (see Figure B-1). The processors in one layer are usually fully connected with the processors in the immediate neighboring layers. A processor is sometimes connected to itself and to other processors in its own layer. The connections between processors have associated weights that modify data flowing through the connections. Each processor executes (in parallel with the other processors in its own layer) a weighted summation or product of its input data elements, an intermediate non-linear transfer function, and an output function. The "program" of a neural network is contained in the inter-processor connection weights. There can be hundreds of thousands of these connection weights.

There are many methods for setting the connection weights (sometimes called training the network). Not all methods are logistically supportable. (See Chapter 1. Logistical supportability refers to modifiability and maintainability in the face of changing requirements.) All methods involve sending "training" data through the network. These data represent examples of the task the network is to accomplish. As the training data pass through the network, the weights are adjusted automatically according to one of several training methods. The network will respond appropriately to the training data given that the network has had a sufficient exposure to the training data. If the training data adequately represent the task to be accomplished, the network also can correctly process test data that it has not been trained on.

Neural networks are not programmed in the traditional sense. Rather, they adjust themselves to the task at hand based on examples. Computers programmed via traditional sequential methods learn from algorithms composed of explicit task-accomplishment instructions. Neural networks are heuristic in nature, not algorithmic. Because of this, training a neural network is not as straightforward as it might first appear. There are many training methods and many network architectures. Depending on the task at hand, a given training method and network architecture may or may not be appropriate.

The training and performance reliability of a neural network is of primary concern for logistical and mission support reasons. Thus, it is necessary to use a neural network architecture and training method that has predictable training and performance characteristics. Such a network is the exponential neural network discussed in this report.

Klimasauskas et al [1], Lippman [2], Rogers et al. [3], and Rumelhart & McClelland [4] have all written more detailed introductions to neural networks. Lippman's paper is easy to follow and is widely referenced. The other authors have produced full-length books. Klimasauskas and Rumelhart & McClelland provide IBM-PC disks with example networks. For increasing length and level of detail start with Lippmann then go on to Rogers. Follow up with Klimasauskas. Rumelhart & McClelland is the most theoretical of the four.

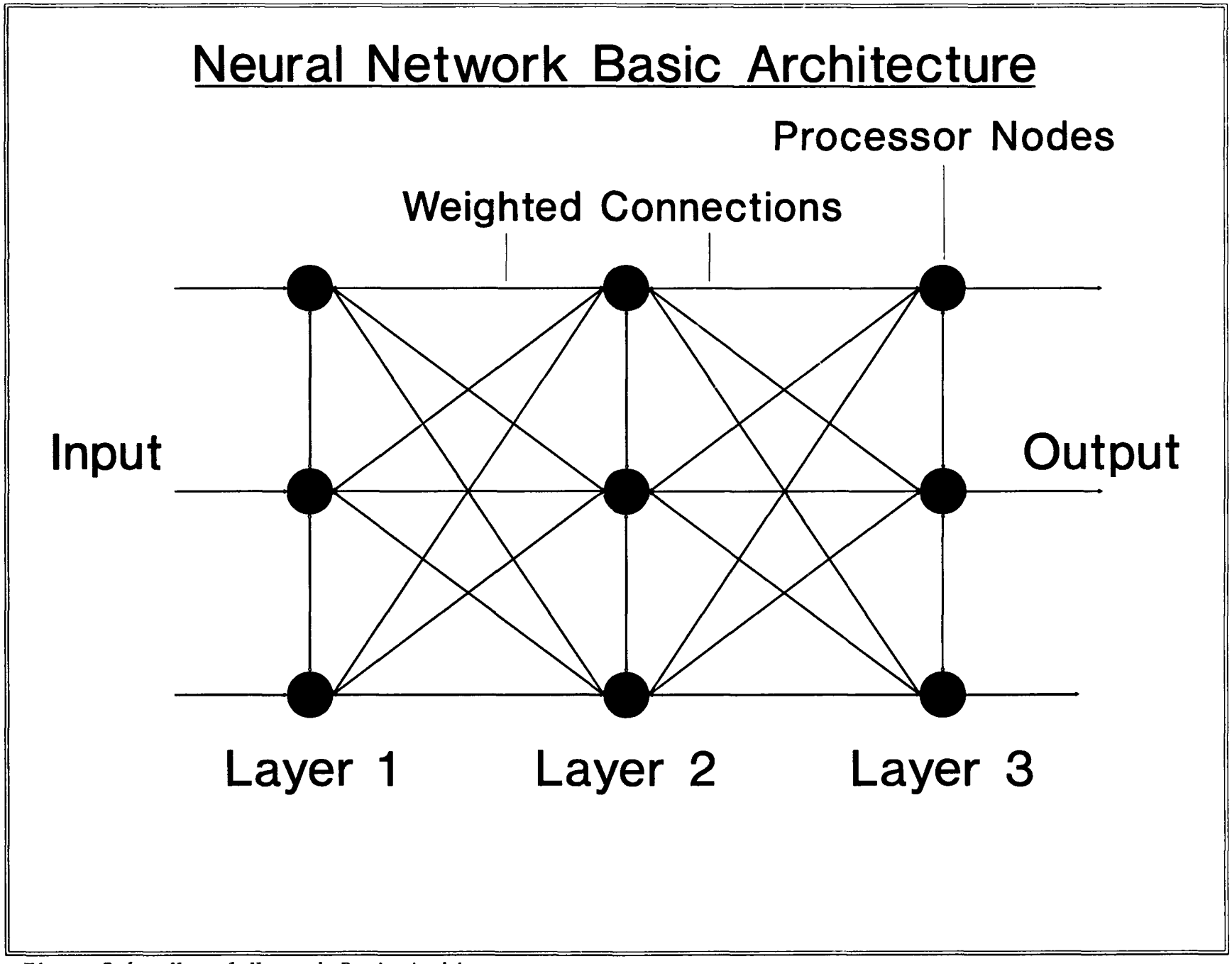

Figure B-1: Neural Network Basic Architecture

Appendix B References

1. Klimasauskas, Casimir; John Guiver; and Garrett Pelton Neural Computing Pittsburgh, PA: NeuralWare, Inc; 1989

2. Lippmann, Richard P. "An Introduction to Computing with Neural Nets", IEEE ASSP Magazine, Apr 87

3. Rogers, Steven; Matthew Kabrisky; Dennis Ruck; and Gregory Tarr An Introduction to Biological and Artificial Neural Networks Bellingham, WA: Optical Engineering Press, 1991

4. Rumelhart, David E. and James L. McClelland Parallel Distributed Processing; Vol I, II, III Cambridge, MA: MIT Press, 1986

(This page intentionally left blank.)

APPENDIX C

AUTHOR BIOGRAPHY

Peter G. Raeth received the B.S. degree in Electrical Engineering from the University of South Carolina in 1979 and the M.S. in Computer Engineering from the Air Force Institute of Technology in 1980. He is a Major on active duty with Headquarters Air Force Materiel Command (Science & Technology) and is stationed in Dayton, Ohio. In this capacity, he spearheads an initiative for integrating Air Force planning with industry research. This effort will leverage 10,000 industrial research projects, each with an average value of $500,000. His prior responsibilities, while stationed with USAF Wright Laboratory, focused on conducting, managing, and transitioning research in advanced information processing methods for defense embedded computers. He has published professional articles, papers, and reports on the subjects of expert systems and neural networks since 1977. His book, "Expert Systems: A Software Methodology for Modern Applications", a structured collection of papers, was published by the IEEE Computer Society in 1990. In 1991 he was awarded a research fellowship in neural networks at the University of Dayton Research Institute. He is a member of NeuralWare's product review team and a member of the proposal review team for Academic Press. Major Raeth is a member of Tau Beta Pi, Eta Kappa Nu, Omicron Delta Kappa, IEEE Computer Society, and the Dayton Special Interest Group on Artificial Intelligence. The author may be contacted at HQ AFMC/STAI; WPAFB, OH 45433-5001 USA. Phone: 513-257-5366